Primeros pasos en LOGO

M. G. MONTEIL

PARANINFO SA

Primeros pasos en LOGO

Marie-Gaëlle MONTEIL

Primeros pasos en LOGO

1985

PARANINFO S.A.

MADRID

Traducido por:
Mª JOSE RELINQUE ELETA

© Editions EYROLLES, París (Francia)
© de la edición española, Editorial Paraninfo, S.A., Magallanes, 25 - Madrid-15
© de la traducción española, Editorial Paraninfo, S.A., Magallanes, 25 - Madrid-15

Título original:
PREMIERS PAS EN LOGO

Reservados los derechos para todos los países de lengua española. Ninguna parte de esta publicación, incluido el diseño de la cubierta, puede ser reproducido, almacenado o transmitido de ninguna forma, ni por ningún medio, sea éste electrónico, químico, mecánico, electro-óptico, grabación, fotocopia o cualquier otro, sin la previa autorización escrita por parte de la editorial.

IMPRESO EN ESPAÑA
PRINTED IN SPAIN

ISBN: 84-283-1347-4

Depósito Legal: 39827-1984

PARANINFO SA Magallanes, 25 - 28015 MADRID (3-3363)

Impreso en Lavel. Los Llanos, nave 6. Humanes (Madrid)

Indice

Introducción 9

1. Primer contacto 11

2. Los procedimientos y las variables 15
1. Introducción 15
2. Recordando el BASIC 15
3. Algunas definiciones 16
3.1. Los procedimientos 16
3.2. Comandos, operaciones y primitivos 17
3.3. Las entradas y salidas 18
3.3.1. Las entradas 18
3.3.2. Las salidas 18
4. Las variables 20
4.1. Los comandos DA y NOMBRA 20
4.2. Las variables locales y globales 22
5. Recursividad 22

3. Las palabras y las listas 27
1. Los objetos LOGO 27
2. Las palabras 28
3. Las listas 33
4. Algunos ejemplos 37

4. Los números 39
1. La noción de número 39
2. Los diferentes tipos de números 40
3. Las operaciones matemáticas infijadas 40
4. Las operaciones aritméticas prefijadas 43
5. Las operaciones lógicas 46

5. Los gráficos en LOGO y la tortuga gráfica 48
A. Los primitivos gráficos LOGO 49
1. Los modos texto y gráfico 49
2. La pantalla 50
2.1. Su color 50
2.2. El campo de la tortuga 51
3. El lápiz y la goma 52
3.1. El color del lápiz 52
3.2. La goma 52
4. Los primitivos de desplazamiento de la tortuga 54
5. La posición de la tortuga 55
B. Algunas ideas para que Vd. realice sus propios gráficos en LOGO 56
1. Introducción 56
2. Los triángulos 56
3. Poliedros 59
4. Círculos 61
5. Una mariposa 62

6. Otros procedimientos útiles 66
1. El control de ejecución de los procedimientos 66
2. La visualización 71
3. Los ficheros 72
3.1. Generalidades 72
3.2. La sintaxis 73

7. Programar en LOGO 74
1. Introducción 74
2. Algunos procedimientos aritméticos 74

3. Un programa de cambio de base 77
4. Algunos dibujos 80
5. Algunos ejemplos de programas interactivos 82

8. Glosario de los primitivos LOGO 90
A. Las diversas implementaciones de LOGO 90
B. La sintaxis LOGO 91
1. Los primitivos gráficos 91
2. Los números 92
3. Las palabras y las listas 93
4. Procedimientos y variables 94
5. Otros procedimientos 95

Introducción

LOGO nació alrededor del año 1968 en los laboratorios de Inteligencia Artificial del MIT en los Estados Unidos. Luego ha sido utilizado con éxito en escuelas y universidades, y ahora está disponible en microordenadores tales como Apple II, Apple IIe, TI 99.

Actualmente anunciado en el Commodore 64, el ZX Spectrum, el TO 7 y el IBM PC, su difusión es mayor día a día.

Por su concepción es un lenguaje que resulta accesible a todos.

Muy fácil, y rápidamente asimilable, abre a todos las puertas de una programación cómoda, rica y muy completa.

LOGO posee además dos cualidades esenciales que hacen de él un excelente lenguaje de programación:

- de un lado, permite sin dificultad la manipulación de cadenas de caracteres, texto y símbolos...
- de otro, posee un verdadero modo gráfico que permite la realización de dibujos muy sofisticados.

Esta obra se propone responder a las tres preguntas siguientes:

- ¿Qué es LOGO?
- ¿Qué se puede hacer con LOGO?
- ¿Cómo programar en LOGO?

Un capítulo de introducción les iniciará en el lenguaje y su utilización.

El capítulo 2 está reservado a los procedimientos, núcleos de base de un programa (cómo definirlos y utilizarlos para realizar programas complejos).

Los capítulos 3 y 4 describen los datos LOGO (números, palabras y listas).

El capítulo 5 está reservado a los gráficos; otro capítulo presenta algunas ideas de programas que salen, en parte, del dominio de Inteligencia Artificial.

Esta obra no quiere ser un manual de referencia para tal o cual máquina. Sin embargo, al final les ofrecemos un glosario, con el fin de concretar las diferencias de sintaxis entre las distintas implementaciones de LOGO.

1

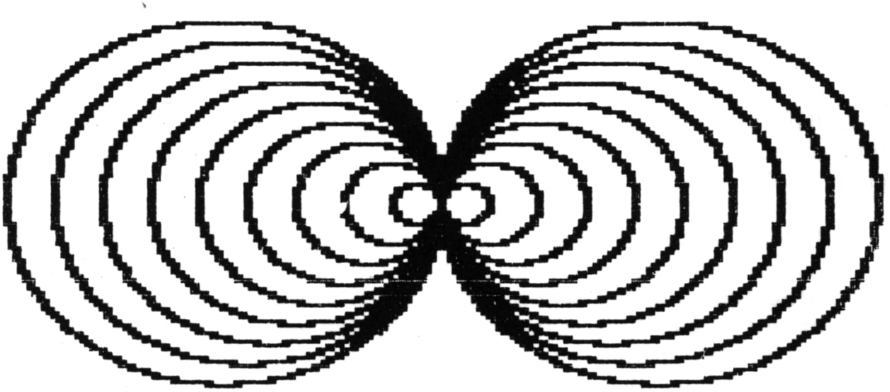
Primer contacto

En primer lugar, este libro no se limita a un curso de LOGO específico para una máquina determinada. La sintaxis que hemos adoptado es relativamente general. Para mayor claridad, los comandos y operaciones han sido descritos en español, con su equivalente en inglés.

Un glosario concreta, al final de esta obra, las sintaxis propias de cada máquina.

Contrariamente al lenguaje Basic que podría desalentar al programador principiante en cuanto quiere escribir un programa poco original, LOGO permite crear muy fácilmente programas interactivos; es posible, por ejemplo, sostener un diálogo con el usuario y dibujar sofisticados gráficos, como los que les enseñaremos a hacer en el capítulo 5.

Ejemplo: GCIRCULO 5Ø dará la figura siguiente: (ver capítulo 5).



Aquellos de ustedes que están familiarizados con el lenguaje Basic quizá no hayan apreciado al comienzo de su aprendizaje ciertas obligaciones impuestas por ese lenguaje: por ejemplo la numeración de las líneas, los subprogramas, las bifurcaciones a calcular.

En LOGO no se numeran las líneas, la noción de subprograma no existe y sólo en raras ocasiones se utilizan los bucles.

Hagamos algunos ensayos.

Enchufe su máquina, un mensaje de bienvenida del tipo:

```
BIENVENIDO A LOGO
?
```

aparece en pantalla.

Para calcular el producto de dos números y visualizarlo se introducirá:

```
?ESCRIBE 5 * 2.1
10.5
```

Para que aparezca el mensaje "BUENOS DIAS, BIENVENIDO A LOGO" se introducirá:

```
?ESCRIBE [BUENOS DIAS, BIENVENIDO A LOGO]
BUENOS DIAS, BIENVENIDO A LOGO
```

y para que aparezca la palabra "HOLA" será suficiente introducir:

```
?ESCRIBE "HOLA
HOLA
```

LOGO no exige ninguna aptitud particular para la programación y es muy fácil de utilizar, además posee todas las características de un lenguaje evolucionado y muy completo:

– LOGO es un lenguaje de procedimientos: los programas están constituidos por procedimientos, equivalentes más o menos a subprogramas Basic. Los procedimientos son definidos a partir de comandos LOGO y son independientes los unos de los otros. El usuario puede definir sus propios procedimientos, pudiendo ser utilizado cada uno de ellos en el interior de otro, y así consecutivamente...

Se comunican entre sí gracias a sus entradas y salidas. Pero, en el próximo capítulo, ya veremos con detalle estos puntos.

Por ejemplo:

```
ESCRIBE, +, -, * son procedimientos.
```

– Como Basic, LOGO es un lenguaje interactivo e interpretado, lo que hace fácil su utilización. Además, los diagnósticos de error son más claros y precisos que en Basic. Si usted quiere, por ejemplo, ejecutar en LOGO un procedimiento inexistente, el intérprete le envía un mensaje del tipo:

```
"DESCONOZCO PROCEDIMIENTO. TRUCO"
```

Si usted introduce + 3, el intérprete le envía:

```
" + NECESITA OTRA ENTRADA"
```

– LOGO no manipula sólo números, cadenas de caracteres como Basic, sino también texto, estructuras de datos complejas llamadas "listas". Estas listas pueden ser transformadas por comandos simples, lo que hace de LOGO un lenguaje potente y bien adaptado a todas las operaciones de manipulación de símbolos.

Ejemplo:

```
?DA "FAMILIA [ABUELA [ANA MARIA]]
?ESCRIBE :FAMILIA
ABUELA [ANA MARIA]

?ESCRIBE SALVOPRIMERO :FAMILIA
[ANA MARIA]

:FAMILIA es la lista [ABUELA [ANA MARIA]]
SALVOPRIMERO :FAMILIA es la lista [ANA MARIA]
```

– Otro punto fuerte de LOGO es la existencia de una tortuga gráfica. Se trata de un pequeño robot que puede desplazarse por la pantalla y obedecer a comandos del tipo AVANZA, RETROCEDE, gira a la DERECHA, a la IZQUIERDA...

Así, para trazar una línea de longitud 5Ø, es suficiente teclear AVANZA 5Ø. Para dibujar un cuadrado con un lado de longitud 3Ø, introduciremos:

```
?REPITE 4[AVANZA 3Ø DERECHA 9Ø]
```

Como veremos a continuación, es posible obtener el mismo resultado tecleando:

```
CUADRADO 3Ø
```

Esto, tras haber definido CUADRADO de la forma siguiente:

```
?IM [CUADRADO]
PARA CUADRADO :LADO
REPITE 4 [AV :LADO DR 9Ø]
FIN
```

Ya hemos terminado este capítulo de introducción. Esperamos que estos pequeños ensayos les hayan intrigado lo suficiente como para que les apetezca continuar con este libro.

El próximo capítulo está dedicado a los procedimientos y a las variables: enseña cómo definir y utilizar los procedimientos para realizar programas LOGO.

2

Los procedimientos y las variables

1. INTRODUCCION

En el capítulo precedente hemos visto que, gracias a ciertos procedimientos, se puede dibujar un cuadrado, calcular la suma, la media de dos números y hacer que los resultados aparezcan en pantalla. En este capítulo, vamos a describir con más precisión cómo definir nuevos procedimientos y cómo utilizarlos para realizar programas.

2. RECORDANDO EL BASIC

En Basic, para efectuar la suma de dos números X e Y cualquiera, se escribirá el programa siguiente:

```
5 INPUT "INTRODUCIR 2 NUMEROS"; X, Y
10 M = (X + Y) / 2
15 PRINT "MEDIA = "; M
20 END
```

El programa se pondrá en funcionamiento pulsando el comando RUN. El mensaje "INTRODUCIR 2 NUMEROS" aparece entonces en

pantalla. Tras haber introducido los dos números por teclado y haber pulsado la tecla RETURN, aparece su media en pantalla:

```
"MEDIA = ...
```

Si desea Ud. conservar tal cual su programa, es preciso guardarlo en casete o en disco y dar entonces un nombre a este programa a fin de poderlo identificar posteriormente.

Por el contrario, si Ud. debe realizar un programa un poco más complicado y sin embargo comenzar por calcular la media de dos números X e Y en un momento dado del programa, construirá un subprograma con las líneas precedentes:

```
 2 REM SUB-PROGRAMA MEDIA
 5 INPUT "INTRODUCIR 2 NUMEROS"; X, Y
10 M = (X + Y) / 2
15 PRINT "MEDIA="; M
20 RETURN
```

El acceso a ese subprograma se efectuará mediante la instrucción Basic GOSUB 2.

En Basic, cada línea comporta instrucciones (REM, INPUT, PRINT, RETURN, ...), operadores (+, /, ...). Existen además los comandos, entre los que está RUN.

X e Y son las variables de entrada del subprograma. Toman necesariamente un valor efectivo (un número real) cuando el programa se ejecuta.

M es la variable de salida del subprograma, también se reemplaza por un número real en el momento de la ejecución.

Observemos, finalmente, que un subprograma se termina mediante la instrucción RETURN y que un programa se termina por END.

3. ALGUNAS DEFINICIONES

3.1. Los procedimientos

Calculemos ahora en LOGO la media de dos números X e Y. Escribiremos:

```
PARA MEDIA :X :Y
ESCRIBE "MEDIA = ESCRIBE/ + :X :Y 2
FIN
```

En LOGO, la noción de programa que llame a subprogramas no existe. Existen únicamente procedimientos, pudiendo cada procedimiento llamar a otro o a sí mismo... Concretaremos esto un poco más adelante.

Las tres líneas que comienzan por PARA... y que terminan por FIN, definen un nuevo procedimiento LOGO, es decir, que gracias a esta sintaxis, el intérprete conoce la existencia del procedimiento MEDIA y sabe, pues, ejecutarlo. Para calcular la media de los números 15 y 30.6, será suficiente introducir en el teclado:

```
?MEDIA 15 30.6
```

La palabra PARA (TO en inglés) indica al intérprete que se quiere definir un nuevo procedimiento. El intérprete le señala entonces que está Ud. en modo definición, haciendo aparecer un signo especial (por lo general >) al comienzo de cada línea.

Igualmente en Basic, cuando escribe Ud. un programa, numera sus líneas.

La palabra FIN (END en inglés) es el equivalente de END en Basic.

Como para el programa Basic, X e Y son las variables de entrada del procedimiento MEDIA o incluso los argumentos del procedimiento MEDIA.

Parece, pues, que todo sucede como si en lugar de escribir subprogramas Basic a los que se accede mediante GOSUB, definiéramos nuevas instrucciones, funciones o comandos Basic.

3.2. Los comandos, las operaciones y los primitivos

Como un comando o una instrucción Basic, un procedimiento realiza una acción. Un procedimiento es, ya un comando, ya una operación, según entregue o no un resultado.

Ejemplo: LIMPIAPANTALLA es un comando que sitúa la tortuga gráfica sola en el centro de una pantalla virgen.

ESCRIBE "palabra es un comando que no da un resultado. En caso necesario, "palabra aparece en pantalla.

MEDIA es una operación, ya que da un resultado de salida (la media de dos números introducidos en el teclado).

Los primitivos son los procedimientos ya implantados en LOGO, ésto por oposición a los procedimientos que define el usuario.

Ejemplo: ESCRIBE, LIMPIAPANTALLA, +, /, ... que acabamos de ver son primitivos. MEDIA no es un primitivo, pues lo hemos definido nosotros mismos a partir de primitivos del lenguaje. Además, es necesario salvaguardarlo en disco (o en casete) si se desea conservarlo.

3.3. Las entradas y salidas

3.3.1. LAS ENTRADAS

Examinemos con más detalle ciertos primitivos: para borrar la pantalla y posicionar la tortuga gráfica en el centro de una pantalla virgen, es preciso introducir el comando LIMPIAPANTALLA (CLEARSCREEN en inglés).

Para trazar una línea de longitud 5Ø, se introducirá AVANZA 5Ø y para una línea de longitud 1ØØ avanza 1ØØ. Contrariamente al comando LIMPIAPANTALLA que efectúa la misma operación cada vez que es llamado, el comando AVANZA número puede efectuar el trazado de rectas de longitudes diferentes. Número es la variable de entrada del procedimiento AVANZA.

Del mismo modo, el procedimiento MEDIA, definido anteriormente, tiene dos variables de entrada X e Y.

Precisemos finalmente que, en el momento de llamada a este tipo de procedimiento, las variables de entrada son reemplazadas por valores efectivos.

Ejemplo:

```
?MEDIA 15 3Ø.6
MEDIA: 22.8?
```

```
?MEDIA -Ø.557 18.22
MEDIA: 8.8315?
?
```

```
?MEDIA 15 3Ø.4
MEDIA: 22.7?
?
```

3.3.2. LAS SALIDAS

Tomemos de nuevo la definición del procedimiento MEDIA:

```
PARA MEDIA :X :Y
MUESTRA "MEDIA: MUESTRA (:X + :Y) /2
FIN
```

y definamos un procedimiento calculando el cuadrado de un número:

```
PARA CUADRADO :X
RETORNA :X * :X
FIN
```

No es posible calcular el cuadrado de la media de dos números utilizando el procedimiento MEDIA, pues éste permite únicamente calcular y mostrar en pantalla la media de dos números X e Y. No proporciona una salida que pueda ser utilizada como entrada por el procedimiento CUADRADO.

Efectuemos ahora:

```
?ESCRIBE CUADRADO 5
25
```

Aquí CUADRADO da un resultado accesible al comando ESCRIBE.

El comando RETORNA variable (en inglés OUTPUT) entrega, saliendo de un procedimiento, resultados accesibles a otros procedimientos. Permite, pues, combinar procedimientos y realizar "programas" complejos.

Este comando, evidentemente, sólo es utilizado en la definición de un procedimiento...

Ejemplo:

```
?ESCRIBE + CUADRADO 3 CUADRADO 5
34
```

El procedimiento CUADRADO es llamado dos veces, los resultados se suman y luego aparecen en pantalla.

```
?MEDIA CUADRADO 3 CUADRADO 5
MEDIA: 17.?
```

Aquí, el procedimiento CUADRADO también ha sido llamado dos veces, entonces la variable X de MEDIA toma el valor 9 (resultado de CUADRADO 3) y la variable Y toma el valor 25.

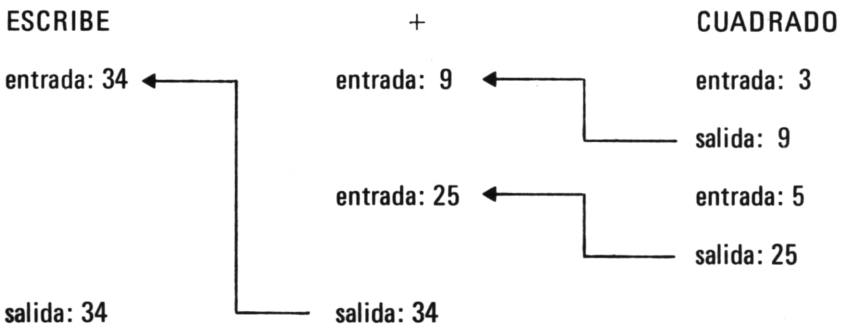
4. VARIABLES

Precisemos un poco cómo se efectúa el paso de argumentos: las variables utilizadas en MEDIA y CUADRADO son variables mudas. En efecto, MEDIA utiliza las variables X e Y y CUADRADO la variable X, pero el hecho de que una misma variable X sea utilizada en los dos procedimientos, no es causa de error.

De hecho, cada procedimiento posee su propia biblioteca de variables y de valores.

Ejemplo:

Concretemos el funcionamiento del intérprete gracias al esquema siguiente:



Así, respecto a CUADRADO 9 y 25 son salidas, respecto a + son entradas.

4.1. Los comandos DA y NOMBRA

Hasta el momento, hemos visto que es posible dar un nombre a un procedimiento, a una variable. Volvamos a tomar el ejemplo del cálculo de la media de dos números: MEDIA es el nombre de este procedimiento. X e Y son los nombres de esas dos variables. Del mismo modo que tenemos la costumbre de nombrar los procedimientos y las variables de un procedimiento, se puede también dar un nombre a un dato gracias a comandos parecidos al LET X = ... en Basic.

El comando DA nombre objeto (MAKE en inglés) establece una relación entre nombre y objeto.

Ejemplo:

```
?DA "NUMERO 5
?ESCRIBE :NUMERO
5
```

5 es la cosa (el objeto) a la que se da el nombre NUMERO. Todo sucede como si se reemplazara el espacio-memoria de nombre NUMERO con el valor 5.

ESCRIBE :NUMERO 5 permite representar en pantalla el contenido del espacio-memoria.

El comando NOMBRA nombre-objeto (NAME en inglés) realiza la misma operación que DA permutando el orden de las entradas.

Ejemplo:

```
NOMBRA 5 "NUMERO
```

tiene como efecto, también, llenar el espacio-memoria de nombre NUMERO con el valor 5.

Existe un comando complementario:

COSA nombre (THING en inglés) permite conocer el contenido del espacio-memoria o celda que se llama nombre.

He aquí algunos ejemplos que les permitirán asimilar mejor estas nociones.

```
?DA "COLOR "NEGRO
?ESCRIBE :COLOR
NEGRO
?DA "FAMILIA [ABUELA [ANA MARIA]]
?ESCRIBE :FAMILIA
ABUELA [ANA MARIA]
?ESCRIBE PRIMERO :FAMILIA
ABUELA
?DA :COLOR [ES BONITO]
?ESCRIBE :COLOR
NEGRO
?ESCRIBE :NEGRO
ES BONITO
```

En la línea DA : COLOR [ES BONITO] asociamos el objeto [ES BONITO] al nombre contenido en la casilla de nombre COLOR. Es decir, se asocia el nombre NEGRO al objeto [ES BONITO].

```
?DA "NOMBRES [MARIA JOSE]
?ESCRIBE :NOMBRES
MARIA JOSE

?ESCRIBE COSA "NOMBRES
MARIA JOSE
```

```
?DA "COLOR "NEGRO
?ESCRIBE :COLOR
NEGRO

?ESCRIBE "COLOR
COLOR
```

4.2. Las variables locales y globales

Ya hemos visto en este capítulo que las variables de entrada de un procedimiento forman parte de una biblioteca propia de este procedimiento. Tomemos de nuevo MEDIA que ya habíamos definido:

```
PARA MEDIA :X :Y
MUESTRA "MEDIA :MUESTRA (:X + :Y) / 2
FIN
```

Si Ud. teclea ESCRIBE :X, LOGO le envía un mensaje de error pues no conoce X. Una variable así, definida solamente en el interior de un procedimiento, es denominada variable local.

Por el contrario, si tecleamos:

```
?NOMBRA 1 "X
?ESCRIBE :X
1
```

LOGO conoce el valor de X que, en este caso, se dice es una variable global.

Estas nociones son sumamente prácticas, pues permiten el paso de parámetros de un procedimiento a otro sin ningún problema...

5. RECURSIVIDAD

Ya hemos dicho que un procedimiento puede llamar en su definición a otro. El procedimiento MEDIA:

```
PARA MEDIA :X :Y
MUESTRA "MEDIA :MUESTRA (:X + :Y)/2
FIN
```

se define con la ayuda de los primivitos del lenguaje LOGO: ESCRIBE, +, / a quienes llama en el curso de su ejecución.

Realicemos ahora un pequeño programa contador:

```
PARA CONTADOR :N
ETIQUETA "BUCLE
SI :N = Ø [STOP]
ESCRIBE :N
NOMBRA - :N 1 "N
IR "BUCLE
FIN

?CONTADOR 3
3
2
1
```

Introducimos por el teclado un número entero N que va disminuyendo una unidad cada vez, el programa se detiene cuando N vale cero. El método aquí adoptado es iterativo: una vez que se ha efectuado el tratamiento con el número N, se vuelve a empezar con el número N-1 y así sucesivamente...

El comando IR "BUCLE (en inglés GO "LOOP) es el equivalente del GOTO número línea en Basic...

La línea DA "N – :N 1 realiza la operación Basic LET N = N - 1. Este método, aún siendo correcto, sin embargo, es poco utilizado en LOGO.

Más bien, definiremos el siguiente procedimiento equivalente a CONTADOR:

```
PARA CONT :N
SI :N = Ø [STOP]
ESCRIBE :N
CONT – :N 1
FIN
```

```
?CONT 3
3
2
1
```

El procedimiento CONT se llama a sí mismo y se aplica cada vez a un argumento decrementado en uno. Tal procedimiento, que se autollama, es denominado recursivo. Examinemos un poco cómo se efectúa el desarrollo del programa:

Primera línea:

si : N = Ø [STOP], el intérprete va a la línea siguiente.

Segunda línea:

ESCRIBE :N, aparece 3 en pantalla.

Tercera línea:

CONT – :N 1, el procedimiento CONT se aplica a N – 1, es decir, 2.

Se vuelve a ejecutar CONT con el valor 2:

N es diferente de Ø

2 aparece en pantalla

se aplica CONT a N – 1, es decir, 1.

N es diferente de Ø

1 aparece en pantalla

se aplica CONT a Ø

Finalmente, nos paramos en la línea 1 pues N vale Ø.

He aquí otro ejemplo muy conocido de procedimiento recursivo:

```
PARA FACT :N
SI :N = Ø [RETORNA 1]
RETORNA * :N FACT – :N 1
FIN
```

Precisemos también aquí su desarrollo.

Ejemplo:

```
?ESCRIBE FACT 4
24
```

Línea 1:

N vale 4, se continúa en la línea 2.

Línea 2:

N - 1 vale 3, es preciso valorar 4 * FACT 3.

Cálculo de FACT 3

Línea 1:

N vale 3.

Línea 2:

N – 1 vale 2, es preciso valorar FACT 2 y retornar

4 * (3 * FACT 2).

Cálculo de FACT 2

Línea 1:

N vale 2.

Línea 2:

N – 1 vale 1, hay, pues, que calcular FACT 1 y retornar

4 * (3 * (2 * FACT 1)).

Cálculo de FACT 1

Línea 1:

N vale 1.

Línea 2:

N – 1 vale Ø, hay que calcular FACT Ø y retornar

4 * (3 * (2 * (1 * FACT Ø))).

Cálculo de FACT Ø

Línea 1:

Se devuelve 1, lo que permite la evaluación de

4 * 3 * 2 * 1 * FACT Ø = 4 * 3 * 2 * 1 * 1

= 24

Aquéllos que ya están familiarizados con la noción de pila, asociarán sin dificultad la noción de recursividad a la de "pila". En el cálculo de FACT 4, ha sido preciso, en efecto, memorizar toda una secuencia de acciones a realizar en cada paso dentro de FACT. En el cálculo de FACT 3, ha habido que memorizar 3 * FACT 2 ..., etc.

FACT 4 se deduce de FACT 3, que, a su vez, se deduce de FACT 2, etc... hasta FACT Ø. Se "apilan" las operaciones a realizar. Así, en la etapa FACT 4, memorizamos 4 * FACT 3. Cuando conocemos el valor de FACT 3 (deducido, a su vez, de FACT 2, FACT 1, FACT Ø), podemos calcular 4 * FACT 3.

Los capítulos siguientes utilizan diversos procedimientos recursivos, lo que les permitirá familiarizarse más con este método rápido y elegante.

3

Las palabras y las listas

1. LOS OBJETOS LOGO

Un lenguaje como Basic manipula dos tipos de objetos: los números (enteros o reales) y las cadenas de caracteres gracias a un cierto número de operadores aritméticos (+, –, *, /), lógicos (OR, AND, NOT) y a funciones como LEFT$, RIGHT$, MID$,...

Los objetos LOGO de tipo datos son igualmente números y "texto", pero contrariamente al Basic, que recorta el texto en caracteres para formar cadenas de caracteres, LOGO es capaz de manipular estructuras de datos más complejas. Estas estructuras pueden ser simples (palabras, secuencias de caracteres, números) o más complicadas, pues LOGO es capaz de tratar cadenas, sub-cadenas, concatenarlas, imbricar las unas en las otras...

Estas estructuras de datos son llamadas listas, una secuencia de caracteres se llama palabra. Así:

```
BUENOS DIAS
     X
     3
  LOGO.LOGO
    1.72
```

son palabras.

Una lista es una secuencia de objetos LOGO, siendo cada objeto una palabra o una lista: en consecuencia, una lista es una secuencia de palabras –una secuencia de palabras y de listas– o, incluso, una secuencia de listas.

Sintácticamente, una palabra está precedida de comillas (") y una lista está encuadrada por dos corchetes (< >).

He aquí algunos ejemplos de listas: <BUENOS DIAS BIENVENIDO A LOGO>, <ABC>, <BUENOS DIAS <BIENVENIDO A LOGO> >, <<BUENOS DIAS> <BIENVENIDO <A LOGO>>>, <1 2 3>.

La lista <BUENOS DIAS BIENVENIDO A LOGO> contiene cinco elementos (cinco palabras).

<BUENOS DIAS <BIENVENIDO A LOGO>> contiene tres. El primer elemento es la palabra BUENOS, el segundo DIAS y el tercero es una lista de tres palabras.

<<BUENOS DIAS> <BIENVENIDO < A LOGO>>> contiene dos. El primer elemento <BUENOS DIAS> es una lista, el segundo elemento: <BIENVENIDO <A LOGO>> es una lista que contiene a su vez dos elementos: la palabra BIENVENIDO y la lista <A LOGO>.

La palabra vacía se indica " y la lista vacía <>.

2. LAS PALABRAS

Al considerarse a los números como palabras LOGO, existen primitivos específicos para el tratamiento de las palabras LOGO que se aplican igualmente a los números. Tomemos el ejemplo del procedimiento PRIMERO: PRIMERO "123 (en inglés FIRST "123) tiene como resultado 1. Del mismo modo, PRIMERO "ABCD es igual a A.

En este capítulo estudiaremos este tipo de procedimientos, comunes a todas las palabras LOGO; los primitivos propios del cálculo científico serán examinados ulteriormente.

Existen algunas analogías entre ciertos primitivos LOGO y ciertas funciones Basic; así es posible obtener el código ASCII de un carácter tecleado, e, inversamente, saber cuál es el carácter asociado a un código ASCII dado.

La operación ASCII carácter devuelve el código ASCII de "carácter".

Ejemplo:

```
?ESCRIBE ASCII "A
65
```

La operación inversa, CAR número (en inglés CHAR number) da el carácter cuyo código ASCII es el número especificado.

Ejemplo:

```
?ESCRIBE CAR 79
O
```

La operación Basic de concatenación (suma de dos cadenas de caracteres) tiene también su equivalente, en LOGO es el primitivo PALABRA (WORD en inglés).

PALABRA palabra1, palabra2... palabran es una palabra única en la que todas las "palabras " introducidas son colocadas las unas a continuación de las obras.

Ejemplo:

```
?ESCRIBE PALABRA "PALABRA "COMPUESTA
PALABRACOMPUESTA
?ESCRIBE PALABRA "ESTO "ES" "UNA"PRUEBA
ESTO"ES"UNA"PRUEBA
```

Escribamos un procedimiento que nos permita añadir un sufijo a una palabra:

```
PARA SUFIJO :PB
ESCRIBE PALABRA :PB "SUFIJO
FIN

?SUFIJO "PREFIJO
PREFIJOSUFIJO
```

PALABRAP palabra (en inglés WORDP) es un primitivo que permite saber si un objeto LOGO es o no una palabra. El resultado es VERDADERO si es una palabra, FALSO en el caso contrario.

Todos los primitivos de este tipo que dan como resultado VERDADERO o FALSO, se llaman predicados.

Ejemplo:

```
?ESCRIBE PALABRAP 3
VERDADERO
?ESCRIBE PALABRAP "LOGO
VERDADERO
?ESCRIBE PALABRAP [3]
FALSO
?ESCRIBE PALABRAP [ ]
FALSO
?ESCRIBE PALABRAP PRIMERO [LOGO]
VERDADERO
?ESCRIBE PALABRAP SP [LOGO]
FALSO
```

(PRIMERO <LOGO> es igual a LOGO) y (SP <LOGO> es igual a la lista vacía < >).

Otro comando de tratamiento de caracteres es el primitivo LEECAR (READCHAR en inglés, es decir, "lee carácter"). Es el equivalente de la función Basic GET A$ muy utilizada en los programas interactivos: en un momento de la ejecución, el programa se detiene y permanece en estado de espera hasta que el usuario introduce un carácter por teclado. He aquí un ejemplo de procedimiento que utiliza LEECAR:

```
PARA DIALOG
ESCRIBE [INTRODUZCA UN CARACTER POR TECLADO :1 0 2
  Y PULSE RETURN]
SI LEECAR = 1 [ESCRIBE "BUENOS–DIAS STOP]
ESCRIBE "BUENAS–TARDES
FIN

?DIALOG
INTRODUZCA UN CARACTER AL TECLADO : 1 0 2 Y PULSE RETURN
BUENOS–DIAS
```

Describamos ahora las operaciones de tratamiento de cadenas de caracteres similares a LEN, MID$, LEFT$, RIGHT$.

En primer lugar, el equivalente del LEN en Basic es CUENTA objeto (en inglés COUNT object), siendo objeto una palabra o una lista. Si objeto es una palabra, CUENTA objeto da el número de caracteres de la palabra. Si objeto es una lista, CUENTA objeto da el número de elementos (palabras o listas) de la lista.

Ejemplo:

```
?ESCRIBE CUENTA [LOGO ES UN LENGUAJE]
4

?ESCRIBE CUENTA [LOGO [ES UN [LENGUAJE]
3

?ESCRIBE CUENTA [LOGO]
1
```

Las cuatro operaciones siguientes: PRIMERO, SALVOPRIMERO (SP), ULTIMO, SALVOULTIMO (SU) se aplican igualmente a las palabras y a las listas. Sin embargo, esos elementos deben ser no vacíos. (Si es una palabra, que sea diferente de la palabra vacía anotada", y si es una lista que sea diferente de la lista vacía anotada <>).

PRIMERO objeto (FIRST object en inglés) da el primer elemento de objeto: si objeto es una palabra, entrega un carácter –si objeto es una lista, PRIMERO lista entrega ya sea una palabra, ya sea una lista.

Ejemplo:

```
?ESCRIBE PRIMERO [LOGO ES UN LENGUAJE]
LOGO
?ESCRIBE PRIMERO [LOGO [ES UN LENGUAJE]]
LOGO
?ESCRIBE PRIMERO [[LOGO] ES UN LENGUAJE]
LOGO
```

SALVOPRIMERO objeto (BUTFIRST object en inglés) tiene como resultado el objeto privado de su primer elemento. Si objeto es una palabra, se suprime el primer caracter –si es una lista, se suprime el primer elemento de la lista.

Ejemplo:

```
?ESCRIBE SP [PEDRO COME PAN]
COME PAN
```

el primer elemento de la lista es la palabra PEDRO.

```
?ESCRIBE SP [PAN]
```

la lista tiene un solo elemento, de ahí el resultado <>.

```
?ESCRIBE SP "PAN
AN
```

objeto es aquí una palabra de tres elementos.

Ciertos procedimientos complementarios de estos últimos tratan al objeto comenzando por el final de éste.

ULTIMO objeto (en inglés LAST) da el último elemento de objeto. Así, si objeto es una palabra, LAST objeto es la última letra de la palabra.

Ejemplo:

```
?
?ESCRIBE ULTIMO [DOGS]
DOGS
```

Se aplica ULTIMO a una lista de un elemento.

```
?ESCRIBE ULTIMO "DOGS
S
```

Se aplica ULTIMO a una palabra.

```
?ESCRIBE ULTIMO [DOGS [CATS]]
CATS
```

Se aplica ULTIMO a una lista de dos elementos de los que el segundo es una lista.

SALVOULTIMO objeto (en inglés BUTLAST) priva a objeto de su último elemento.

Ejemplo:

```
?ESCRIBE SU [DOGS]
```

SALVOULTIMO (SU) se aplica a una lista de un elemento.

```
?ESCRIBE SU[DOGS [CATS]]
DOGS
```

Aquí SU se aplica a una lista de dos elementos.

Para completar estas cuatro operaciones de manipulación de estructuras de datos, utilizables solamente con palabras y listas no vacías, se utiliza con mucha frecuencia el predicado VACIOP objeto (EMPTYP object en inglés).

VACIOP objeto da el resultado VERDADERO si objeto es la palabra vacía o la lista vacía – VACIOP puede, por ejemplo, ser aplicado antes de ejecutar los primitivos PRIMERO, SP, ULTIMO, SU.

El procedimiento IGUALP objeto1 objeto2 ha sido alguna vez implantado en ciertos intérpretes. Equivalente a la igualdad clásica (=), permite preguntar la igualdad de dos números, palabras o listas. También es un medio de saber si dos objetos son del mismo tipo (tipo número, palabra o lista...).

Ejemplo:

```
?ESCRIBE IGUALP [DOGS] "DOGS
FALSO
```

el primer objeto es una lista, el segundo una palabra.

Observen las diferentes sintaxis de IGUALP y del primitivo =. Escribiremos: IGUALP 1 2 y 1 = 2, la primera expresión adopta la anotación prefijada (el operador precede a los datos) mientras que la segunda expresión adopta la forma clásica. Esto se describe con más detalle en el capítulo que trata de los números.

3. LAS LISTAS

Ya hemos descrito los primitivos PRIMERO, SALVOPRIMERO, ULTIMO, SALVOULTIMO y los predicados IGUALP y VACIOP.

Existe un primitivo que, aplicándose a listas, realiza casi la misma función para las listas que PALABRA para las palabras.

LISTA objeto 1 objeto 2 objeto 3... objeton (en inglés LIST) crea una lista que tiene por elementos objeto1, objeto2, ... objeton.

```
?ESCRIBE LISTA "CAT "DOG
CAT DOG
```

Se aplica LISTA a dos palabras: se obtiene, pues, una lista "lisa" de dos palabras. (Se llama lista lisa a una lista cuyos elementos son palabras.)

Ejemplo:

```
?ESCRIBE LISTA "CAT ["DOG "PIG]
CAT ["DOG "PIG]

?ESCRIBE LISTA "CAT [DOG PIG]
CAT [DOG PIG]

?ESCRIBE LISTA "A [ ]
A [ ]
```

Del mismo modo que READCHAR manipula caracteres, existe un comando READLIST que permite al usuario introducir un texto por teclado.

Los siguientes procedimientos son propios de las listas y permiten realizar acciones mucho más evolucionadas y complejas de lo que permiten lenguajes clásicos como FORTRAN, BASIC... Las operaciones FPUT, LPUT, SENTENCE permiten la concatenación de listas, la inserción, truncamiento de sub-cadenas en el interior de una cadena, etc...

FPUT object list (en español PONP) realiza la inserción del objeto al comienzo de la lista. Efectúa más o menos la operación contraria a SALVOPRIMERO, que suprime el primer elemento de una lista. Podríamos caer en la tentación de comparar FPUT con LISTA: estos dos procedimientos operan, sin embargo, de forma diferente, notablemente al nivel de los paréntesis. LISTA opera sobre cualquier objeto mientras que FPUT no puede tener más que dos elementos, el segundo de los cuales es necesariamente una lista. (Podemos igualmente observar que el número total de paréntesis dentro de objeto y lista es el mismo que en el resultado de FPUT.

Ejemplo:

```
?ESCRIBE PONP "PERRO [GATO CERDO]
PERRO GATO CERDO
```

Aquí el objeto es la palabra "PERRO y lista comporta dos elementos; el resultado es una lista de tres elementos.

```
?ESCRIBE PONP [PERRO] [GATO CERDO]
[PERRO] GATO CERDO
```

Aquí objeto es la lista <PERRO> y lista comporta dos palabras; el resultado es una lista de tres elementos de los que el primero es la lista <PERRO>.

El primitivo SENTENCE permite concatenar diferentes objetos. SENTENCE objeto1 objeto2... objeton (en español FRASE) se aplica a un número cualquiera de objetos (palabras o listas) y forma una lista lisa compuesta de todos esos objetos. Así:

```
FRASE palabra1 palabra2 dará [palabra1 palabra2]
FRASE palabra1 [lista2] dará [palabra1 lista2]
```

Por el contrario,

```
LISTA palabra1 palabra2 dará [palabra1 palabra2]
```

y

```
LISTA palabra1 lista2 dará [palabra1 [lista2]]
```

Tampoco el número de corchetes es el mismo cuando se emplea LISTA o FRASE.

Ejemplo:

```
?ESCRIBE FRASE "CAT "DOG
CAT DOG
```

LISTA da el mismo resultado.

```
? ESCRIBE FRASE [UNA FRASE] [ES [UNA FRASE] ]
UNA FRASE ES [UNA FRASE]
```

```
?ESCRIBE LISTA [UNA LISTA] [ES [UNA LISTA]]
[UNA LISTA] [ES [UNA LISTA]]
```

La operación LPUT object list (en español PONF) inserta el objeto al final de la lista. Es más o menos complementaria de SALVOULTIMO, cuyo papel es suprimir el último elemento de una lista.

Ejemplo:

```
?ESCRIBE PONF "PERRO [GATO CERDO]
GATO CERDO PERRO

?ESCRIBE PONF [PERRO] [GATO CERDO ]
GATO CERDO [PERRO]
```

Otra operación es también muy útil:

MIEMBROP objeto lista permite cuestionar la pertenencia de un objeto a una lista. MIEMBROP es un predicado, pues da los valores VERDADERO y FALSO. Tiene, necesariamente, dos argumentos, el primero es cualquiera, el segundo tiene que ser una lista. Si objeto es elemento de lista, MIEMBROP da el valor VERDADERO y el valor FALSO en caso contrario.

Ejemplo:

```
?ESCRIBE MIEMBROP "A [1 2 A C B]
VERDADERO
```

lista comporta cinco palabras una de las cuales es la palabra A.

```
?ESCRIBE MIEMBROP 3 [1 2 [3 4 5]]
FALSO
```

lista comporta tres elementos, el primero es 1, el segundo es 2, el tercero es la lista [3 4 5].

```
?ESCRIBE MIEMBROP [1 2] [1 2 [3 4 5]]
FALSO
```

lista comporta los tres elementos enumerados anteriormente y objeto es la lista [1 2].

4. ALGUNOS EJEMPLOS

Para precisar mejor las ideas, he aquí algunos programas simples de aplicación de los procedimientos que hemos presentado. En primer lugar, tenemos un divertido procedimiento que concatena una palabra a sí misma.

```
?IM [DOBLEP]
PARA DOBLEP :PB
ESCRIBE PALABRA :PB :PB
FIN

?DOBLEP "LOGO
LOGOLOGO
```

Esto sólo se aplica a una palabra y no a una lista: para evitar errores, se puede preguntar si se introduce efectivamente bien una palabra modificando un poco el procedimiento anterior.

```
PARA DOBLEP :PB
SI NO PALABRAP :PB [STOP]
ESCRIBE PALABRA :PB :PB
FIN

?DOBLEP [QWERTY]
```

Escribiremos para las palabras un procedimiento que vaya quitando un carácter a una palabra.

```
PARA REDUCIRP :PB
SI NO PALABRAP :PB [STOP]
ESCRIBE :PB
SI IGUALP :PB " [STOP]
REDUCIRP SP :PB
FIN

?REDUCIRP "LOGO
LOGO
OGO
GO
O

?REDUCIRP [ASD]
```

Este procedimiento se aplica recursivamente a la palabra PB introducida privándola de un carácter a cada llamada del procedimiento. Podemos, igualmente, reescribir procedimientos para aplicarlos a listas.

```
PARA DOBLEL :L
SI NO LISTAP :L [STOP]
ESCRIBE LISTA :L :L
FIN

?DOBLEL "ASD

?DOBLEL [LOGO]
[LOGO] [LOGO]

PARA REDUCIRL :L
SI NO LISTAP :L [STOP]
ESCRIBE :L
SI IGUALP :L [ ] [STOP]
REDUCIRL SP :L
FIN

?REDUCIRL [A[B[C]H]]
A [B [C] H]
[B [C] H]

?REDUCIRL [QWERTY]
QWERTY
```

Hemos terminado con las primitivas corrientes de manipulación de palabras y listas. Más adelante abordaremos ciertos procedimientos más complejos que permiten usos más específicos de listas.

4

Los números

En el capítulo reservado a las palabras y a las listas, hemos visto que los números son palabras LOGO y, por lo tanto, que los primitivos que manipulan palabras son igualmente aplicables a los números. Vamos a describir ahora otros primitivos específicos para los números.

1. LA NOCION DE NUMERO

En primer lugar, del mismo modo que los predicados PALABRAP y LISTAP (indicando la P la noción de predicado) permiten averiguar si un objeto LOGO es una palabra o una lista, se puede saber si es un número gracias al predicado NUMEROP (NUMBERP en inglés).

Ejemplo:

```
?ESCRIBE NUMEROP "ABC
FALSO

?ESCRIBE NUMEROP [1]
FALSO

?ESCRIBE NUMEROP "1
VERDADERO

?ESCRIBE NUMEROP 1
VERDADERO
```

"ABC es una palabra, <1> es una lista, "1 y 1 son números.

2. LOS DIFERENTES TIPOS DE NUMEROS

Los números son, generalmente, de dos tipos: de tipo entero y de tipo decimal. Así, 3 y 4 son enteros, 3.14 y 3. son decimales que pueden ser indicados bajo forma decimal (como es el caso aquí) o bien bajo forma exponencial.

Escribiremos 314N2 en lugar de 3.14. 314 se llama mantisa y 2 exponente. La mantisa puede ser entera o decimal; el exponente tiene que ser, necesariamente, entero.

La letra N indica un exponente negativo: 314N2 significa 314.10^{-2}.

La letra E indica, por el contrario, exponentes positivos: así 2E6 significa 2.10^{6}.

Hagamos la observación de que en ciertas máquinas (TI 99 por ejemplo), los números son exclusivamente enteros, lo que impide realizar cálculos científicos...

3. LAS OPERACIONES MATEMATICAS INFIJADAS

Ya hemos mencionado las nociones de notación infijada y prefijada. Existe también una notación postfijada. Concretemos un poco: clásicamente, para sumar dos números, a y b se escribe a + b: + es llamado operador, a y b son los argumentos del operador +.

Para efectuar el producto c por a + b escribiremos (a + b) * c. Aquí, todavía, los argumentos del operador * están situados a un lado y otro de éste.

La notación prefijada coloca en primer lugar al operador y posteriormente coloca sus argumentos. Así (a + b) se anotará + a b en prefijado: el operador + es seguido de sus dos argumentos a y b.

Volvamos a tomar el ejemplo de (a + b) * c: el operador * se aplica a c y (a + b), empezamos por escribir * (a + b) c. El primer argumento (a + b) puede todavía ser modificado, como ya hemos visto, se escribe + a b, lo que da finalmente. * + a b c.

El último método llamado notación postfijada consiste en escribir en primer lugar los argumentos de un operador y en segundo lugar el operador en cuestión. Así en postfijado, (a + b) se escribe: a b +. Aquellos que están familiarizados con las calculadoras Hewlett Packard

conocen, con toda seguridad, la notación postfijada también llamada "notación polaca inversa". Las calculadoras Texas Instruments, Casio, etc... trabajan, en cambio, en infijado.

El lenguaje LOGO posee a la vez operaciones infijadas y prefijadas. Sucede incluso, a veces, que una operación infijada tenga su equivalente prefijado.

Describamos en este parágrafo las operaciones infijadas de LOGO.

+:

a + b realiza la adición de los números a y b.

Ejemplo:

```
?ESCRIBE 5 + 3
8
```

-:

a - b realiza la sustracción de los números a y b.

Ejemplo:

```
?ESCRIBE 5.5 – 3
2.5
```

*:

a * b efectúa el producto de a y b.

Ejemplo:

```
?ESCRIBE 14 * 5.5
77.
```

/:

a / b efectúa la división de a y b con b # Ø.

Ejemplo:

```
?ESCRIBE (5 * 3) / 2
7.5
```

El orden en que se efectúan las operaciones es el mismo que en Basic: la multiplicación y la división son prioritarias con respecto a la adición y la sustracción.

```
?ESCRIBE 5 + 3 / 2          ?ESCRIBE (5 + 3) / 2
6.5                         4
```

Las tres operaciones siguientes son tres predicados que permiten comparar dos números. Son muy utilizados en las instrucciones condicionales de los procedimientos para realizar bifurcaciones.

<:

El predicado a < b da el valor VERDADERO si a es menor que b y el valor FALSO en caso contrario.

Ejemplo:

```
?ESCRIBE -7.5 < -10
FALSO

?ESCRIBE 2 < 3
VERDADERO
```

>:

Del mismo modo, el predicado a > b da VERDADERO si a es mayor que b, y FALSO en caso contrario.

Ejemplo:

```
?ESCRIBE -7.5 > -10.1
VERDADERO
```

Si estos dos predicados no tienen equivalente prefijado en su intérprete, es muy simple escribir procedimientos equivalentes:

```
PARA MAYOR :A :B
SI :A > :B [ESCRIBE "VERDADERO]
SI :A < :B [ESCRIBE "FALSO]
FIN
```

```
PARA PMENOR :A :B
SI :A > :B [ESCRIBE "FALSO]
SI :A < :B [ESCRIBE "VERDADERO]
FIN

?PMAYOR 7 5                    ?PMENOR 8 9
VERDADERO                      VERDADERO

?PMAYOR 7 9                    ?PMENOR 7 2
FALSO                          FALSO
```

= :

El último predicado es el predicado = que verifica la igualdad de dos palabras, números o listas.

Ejemplo:

```
?ESCRIBE 100 = 50 *2
VERDADERO

?ESCRIBE 3 = PRIMERO "310
VERDADERO

?ESCRIBE [3] = PRIMERO [310]
FALSO
```

4. LAS OPERACIONES ARITMETICAS PREFIJADAS

Describamos, en primer lugar, los equivalentes prefijados de las operaciones del apartado precedente. SUMA a b es lo mismo en prefijado que (a + b) en infijado. Del mismo modo, PRODUCTO a b y (a * b) son equivalentes.

REDONDEA a tiene como resultado el entero más próximo del número a (aplicado a un entero, da, naturalmente, el entero en cuestión).

ENT a da la parte entera del número a, es decir, el entero inferior o igual a a.

Ejemplo:

```
?ESCRIBE REDONDEA 2.4
2
?ESCRIBE ENT 2.4
2
?ESCRIBE REDONDEA -1.8
-2
?ESCRIBE ENT -1.8
-1
?ESCRIBE ENT -0.98
0
?ESCRIBE REDONDEA 2.6
3
?ESCRIBE ENT 2.6
2
```

Dos operaciones prefijadas se asocian a la división de dos números: se trata de COCIENTE y RESTO. COCIENTE a b es la parte entera de la división de a por b, es decir, el cociente de a por b. RESTO a b da el resto de la división de a por b.

Ejemplo:

```
?ESCRIBE COCIENTE 15 6
2
?ESCRIBE RESTO 15 6
3
```

RAC a es el equivalente del SQR a en Basic: aplicado a números positivos, da su raiz cuadrada.

Ejemplo:

```
?ESCRIBE RAC 23
4.79583
```

Las funciones trigonométricas:

En ciertas máquinas se han implantado también las funciones trigonométricas clásicas tal como SEN a, COS a y, a veces, también ARCTAN a. Recordemos que SEN, COS y ARCTAN tienen por argumento un real cualquiera. ARCTAN da un ángulo como resultado.

Recordemos algunas definiciones trigonométricas:

Si tg X = A entonces ARCTAN A = X (X es el arcotangente de A).

tg X = (sen X) / cos X,

$$\text{arcsen } X = \arctan (X / \sqrt{1 - X^2}),$$

$$\text{arccos } X = \arctan (\sqrt{1 - X^2} / X).$$

SEN A es una operación prefijada que da el seno del número A, COS A da el coseno del número A y ARCTAN A el arcotangente del número A.

Ejemplo:

```
?ESCRIBE SEN Ø
Ø
?ESCRIBE SEN 9Ø
1.
?ESCRIBE SEN 45
.7Ø71Ø7
?ESCRIBE COS Ø
1.
?ESCRIBE COS 9Ø
Ø.
?ESCRIBE ARCTAN 1.8
6Ø.9457
?ESCRIBE ARCTAN 1
45.ØØØ7
```

A partir de estos procedimientos de base podemos definir las operaciones tangente, arcos, arcsen, habida cuenta del repaso trigonométrico efectuado.

```
PARA TANG :X
ESCRIBE / SEN :X COS :X
FIN

PARA ARCOS :X
ESCRIBE ARCTAN/RAC 1 – :X * :X :X
FIN

PARA ARCSEN :X
ESCRIBE ARCTAN/:X RAC 1 – :X * :X
FIN
```

```
?TANG 45
1.
?TANG 89
57.29
?TANG 89.99
5724.27
?ARCOS .ØØ5
89.7135
?ARCOS 1
Ø.
?ARCOS .7Ø7
45.ØØ8
?ARCSEN .7Ø7
44.9919
?ARCSEN .9999
89.1892
?ARCSEN .ØØ5
.286442
```

Existe también en LOGO el equivalente de la función RANDOM en Basic.

ALEATORIO n (RANDOM en inglés) genera aleatoriamente un número entero comprendido en el intervalo (Ø, n–1). Como en Basic, la secuencia de números generados por esta operación es periódica, es decir, que al cabo de cierto tiempo se vuelven a repetir los mismos números. Es posible cambiar esta secuencia de números por el comando FIJALEATORIO (RERANDOM en inglés).

He aquí un ejemplo de utilización del procedimiento ALEATORIO.

```
?IM [ALEAT]
PARA ALEAT :N
SI :N = Ø [STOP]
ESCRIBE ALEATORIO :N
ALEAT – :N 1
FIN

?ALEAT 8
4
Ø
2
3
Ø
1
1
Ø

?ALEAT 4
Ø
2
1
Ø

?ALEAT 5
3
3
1
Ø
Ø
```

5. LAS OPERACIONES LOGICAS

Los tres operadores lógicos OR, AND, NOT que encontramos en todos los lenguajes de programación, también están implantados en LOGO. Los tres se aplican a números booleanos (es decir, VERDADERO y FALSO) y su resultado es también booleano. Son muy utilizados en las operaciones de bifurcaciones condicionales.

O (OR en inglés).

O predicado1 predicado2... predicadon da el valor FALSO si todos los predicados son FALSOS y el valor VERDADERO si al menos uno de los predicados es VERDADERO.

Ejemplo:

```
?ESCRIBE O "VERDADERO "FALSO
VERDADERO
?ESCRIBE O "FALSO "FALSO
FALSO
```

Y (AND en inglés).

Y predicado 1 predicado2... predicadon da el valor VERDADERO si todos los predicados tienen el valor VERDADERO, en caso contrario, tiene como resultado FALSO.

Ejemplo:

```
?ESCRIBE Y "VERDADERO "FALSO
FALSO
?ESCRIBE Y 5 5
5 NO ES NI VERDADERO NI FALSO
```

Aparece un mensaje de error pues 5 no es un predicado.

NO (NOT en inglés).

NO predicado: NO se aplica a un argumento único. Si predicado tiene el valor VERDADERO, NO predicado tiene el valor FALSO y el valor VERDADERO en el caso contrario.

Ejemplo:

```
?ESCRIBE NO "FALSO
VERDADERO
?ESCRIBE NO "A = "X
VERDADERO
```

5

Los gráficos en LOGO y la tortuga gráfica

Como ya habíamos dicho, Ud. puede realizar sus propios gráficos en LOGO, ya sea utilizando primitivos de base o bien creando Ud. mismo sus propios procedimientos.

Para ello, debe Ud. ponerse en modo gráfico e introducir un comando tal como MUESTRATORTUGA, DRAW (en español DIBUJA...).

Desde ese momento, un robot, "la tortuga gráfica" está a su disposición para efectuar el trazado de sus dibujos en pantalla. Esta famosa tortuga conoce ya órdenes y las sabe ejecutar: conoce todos los primitivos gráficos de LOGO y es capaz de aprender y ejecutar a continuación todos los procedimientos que Ud. va a definir.

En ciertas máquinas, la tortuga está simbolizada por un triángulo capaz de girar sobre sí mismo (Apple, ...). En otras, podrá Ud. ver una tortuga más completa, pero el principio es el mismo cualquiera que sea la máquina.

En este capítulo, vamos a describir los comandos y operaciones LOGO disponibles en los microordenadores en que está actualmente implantado (Apple II, Apple IIe, TI 99). Para simplificar, como anteriormente, comentaremos este comando en español, y, luego, al final de esta obra, daremos en un glosario sus equivalentes más próximos en cada tipo de máquina.

Habida cuenta de que actualmente no existe un modelo standard en cuanto a la sintaxis de LOGO, hemos optado por presentarles los diferentes comandos y operaciones por tipo.

Hagamos, igualmente, la precisión de que las posibilidades gráficas de Logo son más o menos parejas a las posibilidades de la máquina que Ud.

posee. Así, en el Commodore 64 y el TI 99, existen "sprites", (duendes) objetos cuya forma, color y velocidad podemos definir, y que pueden desplazarse rápidamente por la pantalla. Los microordenadores Apple II, IIe, en razón de su arquitectura material, no tienen estas posibilidades de animación dinámica.

En este capítulo nos limitaremos a los procedimientos relativos a la tortuga gráfica y disponibles en la mayoría de las máquinas que disponen de LOGO.

A. LOS PRIMITIVOS GRAFICOS LOGO

1. LOS MODOS TEXTO Y GRAFICO

Hasta ahora, hemos dicho que LOGO puede manipular números, texto e, igualmente, efectuar dibujos a su elección. Han debido Uds. notar que para dibujar se utiliza la tortuga LOGO, ese triangulito que puede desplazarse y apuntar en todas direcciones... Esta tortuga está únicamente presente cuando se trabaja en modo gráfico, siendo el otro modo el modo texto. Se utilizan comandos LOGO para seleccionar uno de estos modos.

El modo texto es el seleccionado en cada inicio (al enchufar el microordenador). Es, pues, la opción por defecto; en ese caso toda la pantalla está disponible para mostrar texto. Para pasar al modo gráfico a partir del modo texto se introducirá en el teclado el comando PLENAPANTALLA, MEDIAPANTALLA, DRAW (en inglés dibuja) o bien TELLTURTLE...

Inversamente, si a partir del modo gráfico deseamos ponernos en modo texto, se introduce entonces un comando del tipo TEXTOPANTALLA, NODRAW, NOTURTLE, TEXTSCREEN.

MEDIAPANTALLA se distingue de PLENAPANTALLA en ciertos microordenadores (Apple II, IIe ...) en que existen dos modos que permiten realizar gráficos. En el modo PLENAPANTALLA, toda la pantalla está reservada para mostrar dibujos y todo lo que Ud. introduce en el teclado no aparece en pantalla pero es ejecutado por la máquina. En modo MEDIAPANTALLA, una parte de la pantalla se reserva para mostrar gráficos y la otra (generalmente la parte de abajo) se utiliza para visualizar el texto que Ud. introduce en el teclado.

Existen también comandos que permiten en cada modo borrar la pantalla.

En modo gráfico, un comando del tipo BORRAPANTALLA o CLEARSCREEN permitirá borrar la pantalla y volver a colocar la tortuga en el centro de ésta. La abscisa X de la tortuga, su ordenada Y y la dirección (se dice incluso el "rumbo" de la tortuga) son puestas a cero. Otro comando: LIMPIA borra completamente la pantalla, sólo queda la tortuga en el lugar que estaba anteriormente.

En modo texto, para borrar la pantalla se introducirá un comando tal como: BORRATEXTO, CLEARSCREEN, CLEARTEXT... Este tiene el efecto de borrarla completamente y hacer que vuelva a aparecer el cursor en la parte superior, a la izquierda de la pantalla.

Notemos que, contrariamente al caso de una pizarra que se borra, las informaciones presentes en pantalla antes de la ejecución de este comando no se pierden irremediablemente, al contrario, son almacenadas en el espacio de trabajo que LOGO pone a su disposición en el momento de ser cargado. Es posible visualizar de nuevo esta página-pantalla gracias a comandos específicos de cada máquina.

Por ejemplo, IMTODO (o PRINTOUT ALL) muestra en pantalla los nombres, definiciones de todas las variables y de todos los procedimientos.

Habitualmente, para dibujar, tomamos una hoja de papel, un lápiz y una goma. Si Ud. lo desea escoge una hoja de papel de color, así como un lápiz de color. Puede entonces empezar a dibujar. Si Ud. no está satisfecho de sí mismo, puede borrarlo todo o coger una hoja nueva.

En LOGO ocurre casi lo mismo: Ud. dice a LOGO lo que quiere dibujar, escoge, en todo momento, los colores, y si lo que ha hecho no le gusta, borra la página gracias a los comandos que hemos descrito anteriormente: BORRAPANTALLA, LIMPIA...

Ahora vamos a describir cómo elegir el color de fondo de la pantalla, el del lápiz y cómo borrar los trazos no deseados.

2. LA PANTALLA

2.1. Su color

El color de fondo de la pantalla se determina sencillamente con una instrucción del tipo COLORBACKGROUND (FIJACOLORFONDO en español).

La sintaxis puede ser:

FIJACOLORFONDO número

o

FIJACOLORFONDO nombre de color

estando asociado el número a un color.

Ejemplo:

FIJACOLORFONDO Ø vuelve negro el fondo de la pantalla

2.2. El campo de la tortuga

Puede considerarse a la tortuga como situada en un recinto de dimensiones determinadas. Por definición, se llama campo de la tortuga al espacio en que puede desplazarse. Este espacio puede estar limitado o no a la pantalla, como vamos a ver.

Imagínese su hoja de papel, a veces traza Ud. un cuadro con el lápiz y luego dibuja en su interior. En LOGO es exactamente igual: el comando BARRERA (en inglés FENCE) limita en la pantalla el campo de la tortuga, siéndola imposible salir de él. Si sucede que Ud. se sale del cuadro haciendo, por ejemplo: AVANZA 1ØØØ (o FORWARD 1ØØØ en inglés), LOGO le envía un mensaje de error TURTLE OUT OF BOUNDS!, o TORTUGA FUERA DE PANTALLA!

Si desea Ud. trabajar sin cuadro y dejar que la tortuga se desplace en su espacio de trabajo (la hoja de papel sin cuadro), es preciso introducir el comando VENTANA (en inglés WINDOW). El comando precedente AVANZA 1ØØØ no produce entonces mensaje de error, pero la tortuga desaparece de la pantalla. Luego el problema es encontrarla, lo que no siempre es tan fácil...

El último comando del mismo tipo es WRAP (en español RODEA o ENVUELVE). En ese caso, el campo de la tortuga está limitado a la pantalla. Pero si por azar la tortuga atraviesa el borde, LOGO no le envía un mensaje de error.

El campo de la tortuga no es ilimitado. Si Ud. introduce: AVANZA 1ØØØ, la verá Ud. salir por un lado y reaparecer por el otro, y luego detenerse tras haber recorrido la misma distancia.

La pantalla desempeña el papel de una ventana que puede desplazarse para seguir los movimientos de la tortuga.

3. EL LAPIZ Y LA GOMA

3.1. El color del lápiz

Como cuando dibuja, puede Ud. elegir el color de su lápiz en LOGO. Al empezar, la opción por defecto es un lápiz blanco sobre fondo negro. El color del fondo y del lápiz son programables y la gama de colores está en función de su máquina.

Aquéllos que tengan un monitor en blanco y negro, desgraciadamente, no podrán sacar provecho de esta opción color. Podrán obtener matices de gris (o de verde en el caso de un monitor verde).

El comando que permite cambiar el color del lápiz es: FIJACOLOR LAPIZ numero (o bien en inglés PENCOLOR número o PENCOLOR nombre de color – o SETPEN número) según el tipo de máquina.

Ejemplo:

```
FIJACOLORLAPIZ 1 da al lápiz el color blanco
```

3.2. La goma

Cuando se dibuja, es evidente que no se deja siempre la punta del lápiz sobre el papel... Así cuando está Ud. cansado, levanta el lápiz y cuando Ud. hace un trazo, el lápiz está en contacto con el papel.

LOGO ha previsto esas posibilidades:

También la tortuga comprende las órdenes BAJALAPIZ y LEVANTALAPIZ. Después de que se ha introducido un BAJALAPIZ todos los desplazamientos de la tortuga aparecerán en pantalla. Por el contrario, tras el comando LEVANTALAPIZ, si hacemos que la tortuga se mueva, el trazado del recorrido de ésta no aparece.

El ejemplo siguiente permite mostrar un trazo de longitud 5Ø en la pantalla:

```
?BAJALAPIZ
?DR 3Ø
?AV 5Ø
```

Por el contrario, el ejemplo siguiente se contenta con hacer que la tortuga se desplace sin mostrar nada:

```
?LEVANTALAPIZ
?DR 30
?AV 50
```

Para borrar los trazos no deseados, existe una goma. Para borrar, es suficiente volver a pasar sobre los trazos malos. El comando PENERASE (BORRALAPIZ) permite borrar los trazos de lápiz. También, si se hace recorrer a la tortuga gráficos ya existentes, éstos se borrarán de la pantalla.

Otro comando permite borrar y realizar trazos simultáneamente. Es suficiente ejecutar el comando INVERSOLAPIZ (PENREVERSE en inglés). La tortuga se transforma en inversor de color: allí donde se desplaza, los colores del fondo y del lápiz son invertidos. La tortuga traza, pues, líneas donde no las había y borra aquéllas que ya existían.

Ejemplo:

```
PARA TRIANGULO :LADO
REPITE 3 [AV :LADO DR 120]
FIN
```

```
?BL
?TRIANGULO 40
?CENTRO
?IL
?AV 80
?DR 120
?AV 80
?DR 120
```

De este modo, la tortuga borra y traza a la vez.

```
?TRIANGULO 40
?CENTRO
?BORRALAPIZ
?AV 80
```

De este modo, la tortuga se contenta con borrar (borra incluso donde no había nada dibujado).

4. LOS PRIMITIVOS DE DESPLAZAMIENTOS DE LA TORTUGA

Existen en LOGO un cierto número de primitivos correspondientes a instrucciones Basic conocidas (PLOT, DRAW...). Es posible, pues, trazar rectas de pendiente variable, puntos, etc...

En primer lugar, si quiere Ud. permanecer en modo gráfico y no ver continuamente la tortuga en pantalla, es posible hacerla desaparecer introduciendo el comando ESCONDETORTUGA. La tortuga desaparece pero no hemos dejado sin embargo el modo gráfico...

Para que reaparezca la tortuga, es suficiente introducir en el teclado el comando MUESTRATORTUGA (SHOWTURTLE en inglés).

La tortuga puede desplazarse en un espacio de dimensión 2. Se caracteriza por su abscisa X, su ordenada Y, y su rumbo. (Por definición, el rumbo es la dirección a la que apunta la cabeza de la tortuga).

AVANZA número (FORWARD número en inglés). Cuando se introduce esto en el teclado, la tortuga se desplaza una distancia igual al número dado y su trayectoria aparece en pantalla.

El comando RETROCEDE número (BACK en inglés) tiene un efecto similar, pero provoca el desplazamiento de la tortuga en dirección opuesta a su rumbo.

Para hacer girar la tortuga (sea a la derecha, sea a la izquierda), disponemos de los comandos DERECHA ángulo e IZQUIERDA ángulo (respectivamente RIGHT y LEFT en inglés).

Ejemplo: Si el rumbo de la tortuga es 3Ø, será igual a 6Ø tras un comando DR 3Ø.

Disponemos igualmente de comandos que pueden llevar la tortuga a posiciones determinadas de la pantalla. Se trata de FIJAX número y de FIJAY número.

FIJAX número (en inglés SETX) lleva la tortuga a la abscisa número sin modificar su ordenada.

FIJAY número (en inglés SETY) lleva la tortuga a la ordenada número sin modificar su abscisa.

FIJARUMBO ángulo (en inglés SETHEADING ángulo) da a la tortuga la dirección o el "rumbo" ángulo.

Probemos este ejemplo:

```
?BP
?AV 50 DR 90
?FX 0
?FY 0
```

Podemos realizar conjuntamente FIJAX numero1 y FIJAY número2 gracias a un primitivo. Es suficiente con:

FIJAPOSICION <número1 número2> (en inglés SETPOSITION).

Ejemplo:

```
?BP
?AV 50 DR 90
?FPOS [0 0]
```

Esto hace exactamente lo mismo que el ejemplo precedente, pero un poco más rápidamente.

Existe también un comando que permite situar un punto aislado en una posición precisa de la pantalla. Es el comando:

PUNTO <número1 número2> (DOT en inglés) donde número1 es la abscisa del punto que queremos mostrar y número2 su ordenada. El punto es dibujado en la pantalla sin que la tortuga se desplace ni trace una recta entre el lugar en que se encuentra y el punto en cuestión, ello incluso si el lápiz está bajado. (Posición BAJALAPIZ o PENDOWN en inglés).

Ejemplo:

```
?BL
?PUNTO [70 0]
```

5. LA POSICION DE LA TORTUGA

Describamos ahora los comandos y primitivos que indican la posición de la tortuga en la pantalla. Se trata de los primitivos que dan la abscisa X, la ordenada Y y el rumbo de la tortuga.

POSICION da como resultado la abscisa X y la ordenada Y a la vez.

XCOR da únicamente la abscisa X.

YCOR da únicamente la ordenada Y.

Ejemplo:

```
?BP
?AV 50 DR 90 AV 100
?ESCRIBE POS
100. 50
?ESCRIBE XCOR ESCRIBE YCOR
100.
50.
```

RUMBO tiene como resultado un ángulo que indica hacia qué dirección apunta la tortuga. El ángulo es nulo cuando la tortuga apunta al norte.

Ejemplo:

```
?BP AV 50 IZ 100
?ESCRIBE RUMBO
260.
```

B. ALGUNAS IDEAS PARA QUE UD. REALICE SUS PROPIOS GRAFICOS EN LOGO

1. INTRODUCCION

En esta parte vamos a comentar ciertos ejemplos que hemos realizado. Para ello, hemos escrito un cierto número de procedimientos utilizando ya primitivos, ya otros procedimientos previamente definidos.

Ocurre, igualmente, que un procedimiento se llame a sí mismo... Es lo que llamamos un procedimiento recursivo. Algunos procedimientos han necesitado el empleo de primitivos LOGO que quizá les parezcan un tanto misteriosos. Intentaremos comentar en conjunto los ejemplos que les proponemos, pero, para más detalle, les invitamos a dirigirse a los capítulos que tratan esos problemas en concreto.

Estos ejemplos gráficos les son propuestos sobre todo para darles una pequeña idea de lo que se puede hacer con tanta facilidad en LOGO y con tanta dificultad en Basic, por ejemplo...

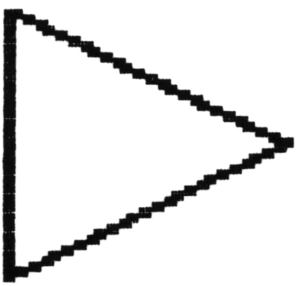
¡ ¡Pero no olviden hacer un BAJALAPIZ antes de empezar!!

2. LOS TRIANGULOS

Un triángulo equilátero tiene tres lados iguales y tres ángulos iguales. Para dibujarlo en pantalla, es suficiente, pues, decir a la tortuga que avance una distancia igual a "LADO", que gire a la derecha 12Ø grados, y esto tres veces. Es conveniente, por tanto, utilizar el comando REPITE, que permite ejecutar la misma operación el número de veces que se determine. (En este caso concreto, la operación se efectúa tres veces).

Así pues, el procedimiento TRIANGULO se define de la manera siguiente:

```
PARA TRIANGULO :LADO
REPITE 3 [AV :LADO DR 12Ø]
FIN
```



Para dibujar el triángulo mostrado, hemos introducido en el teclado TRIANGULO 4Ø, de la misma forma que hubiéramos introducido AVANZA 1ØØ para trazar una recta...

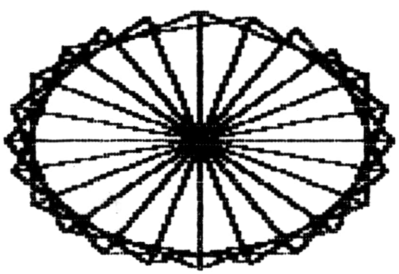
Dibujemos ahora triángulos cada vez más numerosos y ligeramente desplazados los unos de los otros.

```
?IM [PTRIANGULO]
PARA PTRIANGULO :LADO
TRIANGULO :LADO
DR 15
PTRIANGULO :LADO
FIN
```

Este procedimiento empieza, en primer lugar, por trazar un triángulo gracias a TRIANGULO, luego nos desplazamos ligeramente (DR 15), y volvemos a dibujar otro con esos datos y así consecutivamente, indefinidamente... hasta que se detenga la ejecución del programa con STOP o BREAK... según el tipo de máquina que se tenga.

Como han debido Uds. observar, el procedimiento PTRIANGULO es recursivo...

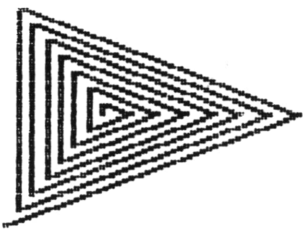
He aquí un ejemplo de ejecución de PTRIANGULO:



Si desea Ud. que se detenga el procedimiento sin tener que hacerlo manualmente con el método indicado anteriormente, convendrá efectuar un test: podemos, por ejemplo, aumentar la longitud del lado de cada nuevo triángulo trazado: (se hace LADO = LADO + 2) y se detiene la ejecución cuando LADO es mayor que 8Ø.

Así, podemos definir el nuevo procedimiento:

```
PARA PTRIANGULO :LADO
TRIANGULO :LADO
DR 15
SI :LADO > 80 [STOP]
PTRIANGULO :LADO + 2
FIN
```



3. POLIEDROS

Dibujemos ahora un conjunto de poliedros regulares. Un poliedro se caracteriza por un lado y un ángulo.

Así, un triángulo equilátero es un poliedro regular con ángulo igual a 120°.

Hemos optado por dibujar no un poliedro simple, sino un conjunto de poliedros imbricados los unos en los otros; para ello hemos escrito el procedimiento siguiente con dos variables (ángulo y lado):

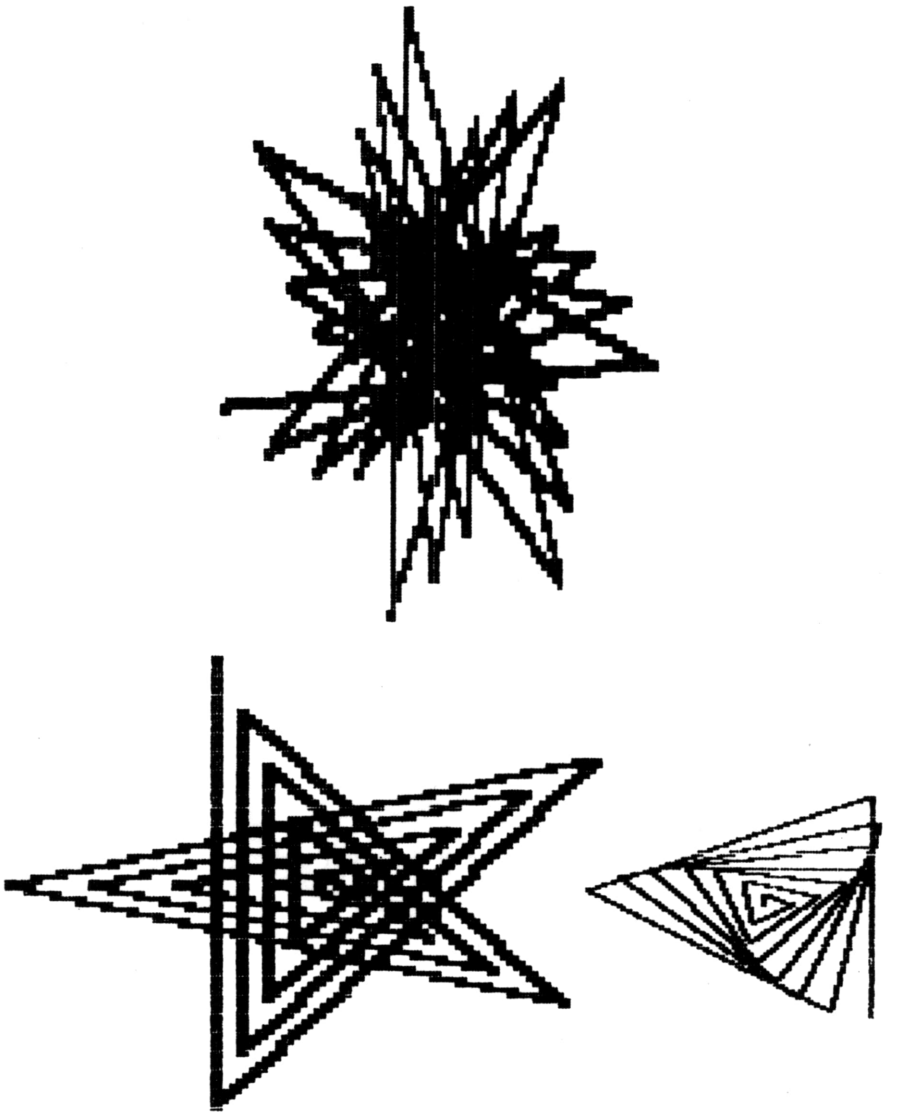
```
PARA POLI :LADO :ANGULO
AV :LADO
DR :ANGULO
POLI :LADO :ANGULO
FIN
```

Podemos, igualmente, aumentar el lado del poliedro y definir el procedimiento siguiente:

```
PARA POLI :LADO :ANGULO
AV :LADO
DR :ANGULO
SI :LADO > 100 [STOP]
POLI :LADO + 2 :ANGULO
FIN
```

Este procedimiento, muy similar a PTRIANGULO, es recursivo; decidimos detener el programa si el lado llega a ser mayor de 1ØØ.

Así hemos efectuado POLI 1Ø 8Ø, POLI 1Ø 12Ø, POLI 1Ø 144, POLI 1Ø 156, POLI 1Ø 16Ø...



4. CIRCULOS

Escribamos en primer lugar el procedimiento CIRCULO1 que permite trazar un círculo: para hacerlo, es preciso girar 36Ø grados. Sea RA un parámetro (diferente al radio del círculo) relativo al "tamaño" del círculo. El procedimiento siguiente traza uno tras otro arcos de círculo que tienen un ángulo en el centro igual a un grado.

```
PARA CIRCULO1 :RA1
REPITE 36Ø [AV :RA1 DR 1]
FIN
```

La circunferencia de este círculo es 36Ø * RA.

Por otra parte, la circunferencia de un círculo es 2 π * radio. de ahí 36Ø * RA = 2 π * radio y RA = 2π * radio / 36Ø.

Este parámetro RA es, pues, igual a: Ø.Ø1745 * radio.

Definamos ahora un procedimiento que tenga como variable el radio del círculo:

```
PARA CIRCULO :RA
CIRCULO1 :RA * 1.745N2
FIN
```

Para trazar varios círculos los unos dentro de los otros, podremos escribir PCIRCULO:

```
PARA PCIRCULO :RA
CIRCULO :RA
SI :RA > 1Ø [STOP]
PCIRCULO :RA 1Ø
FIN
```

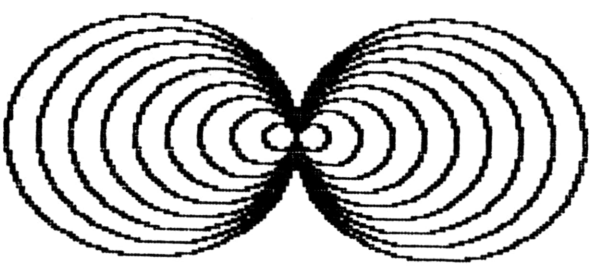
El procedimiento PCIRCULO es también recursivo: en cada pasada, se efectúa un nuevo círculo cuyo radio aumenta.

Para realizar el dibujo de más abajo, podremos escribir tres nuevos procedimientos simétricos de los anteriores, diferenciándose de CIRCULO1, PCIRCULO, CIRCULO por comandos IZQUIERDA en lugar de DERECHA.

El procedimiento ICIRCULO llama a los otros seis (empieza por trazar un conjunto de círculos imbricados unos en otros en la parte derecha de la pantalla, luego efectúa la misma operación en la parte izquierda).

```
PARA ICIRCULO :RA
CT PLENAPANTALLA
PCIRCULO :RA
PCIRCULOI : RA
FIN
```

```
PARA PCIRCULOI :RA
CIRCULOI :RA
SI :RA > 1Ø [STOP]
PCIRCULOI – :RA 1Ø
FIN

PARA CIRCULOI :RA
CIRCULO2 :RA * 1.745N2
FIN

PARA CIRCULO2 :RA2
REPITE 36Ø [AV :RA2 IZ 1]
FIN
```



5. UNA MARIPOSA

He aquí ahora una forma de realizar una mariposa cuya forma podemos modificar ligeramente haciendo variar un parámetro (en este caso V).

Se utilizan dos funciones: FN y CURVA. El procedimiento CURVA borra la pantalla, pone la tortuga en el centro de ésta, oculta la tortuga y llama al procedimiento FN.

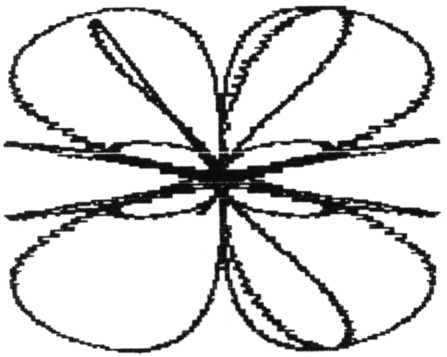
FN sólo tiene un papel de representación: utiliza funciones trigonométricas conocidas por los matemáticos por sus representaciones gráficas tan particulares, luego utiliza FPOS (FIJAPOSICION) primitivo de base con dos argumentos que desplaza la tortuga hasta las coordenadas X e Y especificadas.

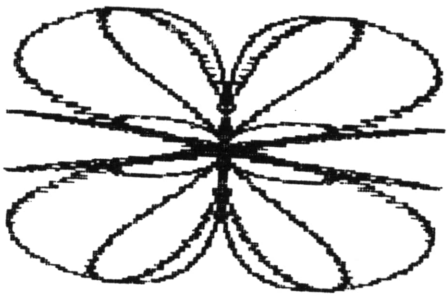
Aquí X e Y son funciones trigonométricas de la variable V que Ud. inicializa al comienzo ejecutando CURVA.

Cuando FPOS ha sido efectuado por la máquina, se vuelve a aplicar FN con un nuevo valor de V (se hace V = V + 1) y así sucesivamente... FN es, pues, recursivo.

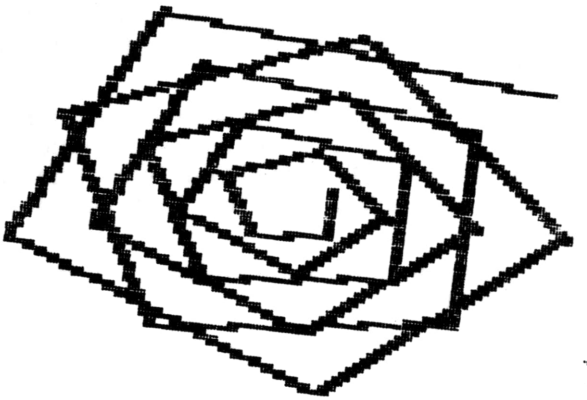
Según el valor de V dado al principio por el usuario, la curva trazada es modificable ligeramente.

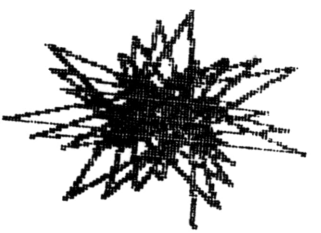
```
PARA CURVA :V
BP CT PLENAPANTALLA FN :V
FIN

PARA FN :V
FPOS LISTA (* 90 SEN * 360 COS / :V 3 COS / V2)
(* 90 COS / :V 3 SEN :V) FN + :V 1
FIN
```

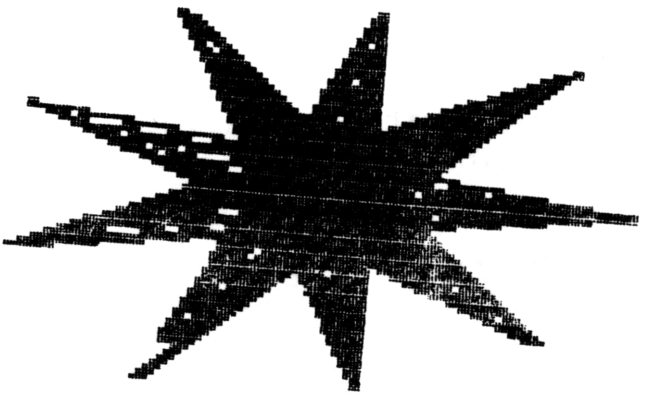


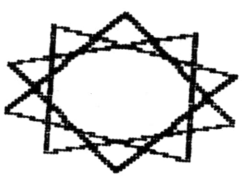


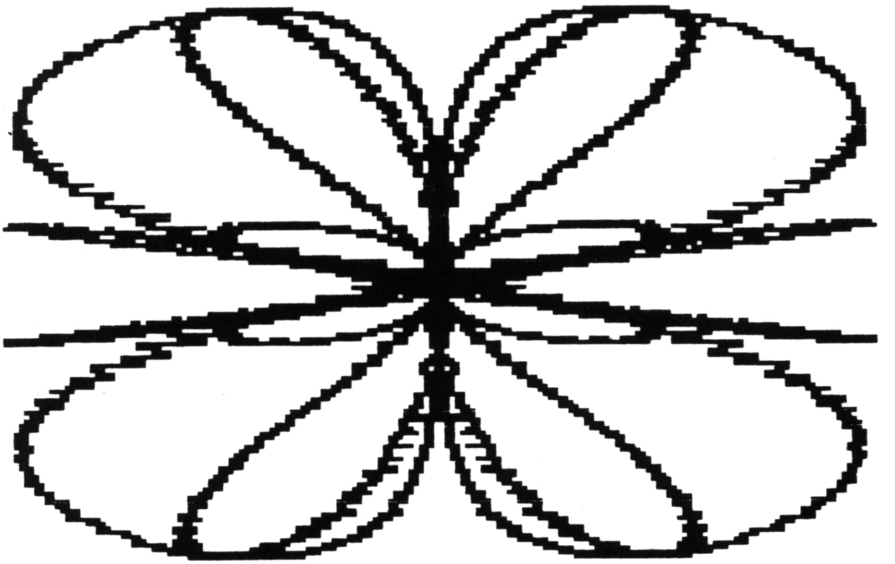
Para terminar, he aquí algunos ejemplos de ejecución de los procedimientos que hemos definido.











Ya hemos terminado con este capítulo. Como se habrán dado Uds. cuenta, las posibilidades gráficas de LOGO son muy importantes y muy variadas según el tipo de máquina que posean.

6

Otros procedimientos útiles

En este capítulo, nos proponemos describir algunos procedimientos muy útiles que les ayudarán a perfeccionar sus programas, así como a salvaguardarlos en disco si así lo desean.

No efectuaremos una descripción detallada de un editor de texto particular ni la de un DOS (sistema de explotación de ficheros). Para más detalle, les aconsejamos se dirijan al manual de referencia de su máquina.

Este capítulo se propone sobre todo concretar algunos puntos generales de importancia.

1. EL CONTROL DE EJECUCION DE LOS PROCEDIMIENTOS

En el capítulo dedicado a los procedimientos y a las variables, tuvimos la ocasión de discutir ámpliamente la forma con que se definen y con la que hacemos que los procedimientos se ejecuten.

En algunos ejemplos propuestos hasta ahora hemos utilizado un cierto número de comandos: ESCRIBE, GIRA, SI..., etc.

Reagrupémoslos y concretemos un poco su empleo.

En primer lugar, para efectuar bifurcaciones, es posible utilizar (según la máquina) tres comandos más o menos similares:

SI (en inglés IF) cuya sintaxis es la siguiente:

SI predicado <exp1>

o

SI predicado<exp1><exp2>

o incluso

IF predicado THEN <exp1>

e

IF predicado THEN <exp1> ELSE <exp2>

Ejemplo:

```
PARA IMP
SI LEECAR = 1 [ESCRIBE "BUENASTARDES]
ESCRIBE "BUENOSDIAS
FIN

?IMP
BUENOSDIAS

?IMP
BUENASTARDES
BUENOSDIAS

PARA IMP
SI LEECAR = 1 [ESCRIBE "BUENASTARDES] [ESCRIBE "BUENOSDIAS]
FIN

?IMP                                    ?IMP
BUENAS TARDES                           BUENOSDIAS
```

Los comandos SIFALSO (IFFALSE en inglés) y SICIERTO (IFTRUE en inglés) van emparejados con el comando PRUEBA.

SICIERTO y SIFALSO (llamados también SIC y SIF respectivamente) realizan o no una acción según el resultado que les proporciona PRUEBA que ha sido efectuado anteriormente.

PRUEBA predicado (REST en inglés) da un resultado cierto o falso según el valor de predicado. Este resultado se conserva para ser utilizado justamente detrás dentro del procedimiento por los comandos SIC y SIF. Observemos que el comando PRUEBA sólo puede utilizarse localmente, en el interior de un procedimiento.

Volvamos a escribir los procedimientos precedentes con PRUEBA, SIC, SIF.

```
PARA IMP
PRUEBA LEECAR = 1
SIC [ESCRIBE "BUENOSDIAS]
SIF [ESCRIBE "BUENASTARDES]
FIN
```

La sintaxis de SIC y SIF es la siguiente:

SIC:

SIC [expresión]. Expresión es ejecutada si el resultado de la última PRUEBA es cierto.

SIF:

SIF [expresión]. SIF ejecuta expresión si el resultado de la última PRUEBA es falso, en caso contrario, no efectúa nada.

Existen otros comandos que también permiten controlar o detener la ejecución de un procedimiento.

Así, el comando STOP detiene la ejecución del procedimiento en curso y vuelve a LOGO o a otro procedimiento. Veamos algunos ejemplos:

```
PARA COUNT :N
ESCRIBE :N
SI :N = Ø [STOP]
COUNT – :N 1
FIN
```

El procedimiento COUNT realiza un pequeño contador que decrementa hasta cero el número N introducido. Cuando se llega a Ø, el procedimiento se detiene.

```
PARA COUNT :N
ESCRIBE :N
SI :N = Ø [STOP]
COUNT – :N 1
FIN

?COUNT 4
4
3
2
1
Ø

PARA CONTAR :N
MUESTRA "DECREMENTAR MUESTRA :N
COUNT :N
ESCRIBE "TERMINADO!
FIN

?
?CONTAR 4
DECREMENTAR44
3
2
1
0
TERMINADO!
```

El procedimiento CONTAR llama al procedimiento COUNT. Cuando la ejecución de COUNT se acaba, puede continuarse la de CONTAR.

PAUSA y CO son dos comandos complementarios muy útiles a la hora de perfeccionar programas.

Como en Basic, PAUSA se utiliza en un procedimiento para detener su ejecución. Se puede continuar tecleando CO (CONTINUA).

Recordemos que las variables de un procedimiento son todas locales y, por consiguiente, difícilmente accesibles.

Así, si queremos conocer en todas las etapas de la ejecución de un procedimiento los valores de las variables que contiene, es preciso parar la ejecución sin salir del procedimiento. Esto es lo que permite hacer el comando PAUSA.

Ejemplo: Volvamos a tomar el procedimiento COUNT:

```
PARA COUNT :N
PAUSA
SI :N = Ø [STOP]
COUNT – :N 1
FIN

?
?
?COUNT 3
EN PAUSA... DENTRO DE COUNT:
PAUSA
COUNT ?:N
NO SE QUE HACER CON 3
COUNT ?CO
EN PAUSA...DENTRO DE COUNT:
PAUSA
COUNT ?:N
NO SE QUE HACER CON 2
COUNT ?CO
EN PAUSA...  DENTRO DE COUNT
PAUSA
COUNT ?CO
EN PAUSA...DENTRO DE COUNT:
PAUSA
COUNT ?:N
NO SE QUE HACER CON Ø
COUNT ?CO
```

PAUSA permite también verificar el buen desarrollo del procedimiento COUNT.

Otros dos comandos utilizados conjuntamente permiten efectuar bifurcaciones. Se trata de VE (GO en inglés) y de LABEL.

Tomemos de nuevo el procedimiento CONTADOR que habíamos definido en el capítulo II.

```
PARA CONTADOR :N
LABEL "BUCLE
SI :N = Ø [STOP]
ESCRIBE :N
NOMBRA – :N 1 "N
VE "BUCLE
FIN
```

```
?CONTADOR 5
5
4
3
2
1
```

LABEL define de alguna forma una etiqueta (en nuestro ejemplo define la etiqueta BUCLE).

Veamos ahora el procedimiento siguiente:

```
PARA CUADRADO :LADO
DA "N Ø
LABEL "BUCLE
AV :LADO DR 9Ø
NOMBRA :N + 1 "N
SI :N = 4 [STOP]
VE "BUCLE"
FIN
```

El procedimiento CUADRADO permite dibujar un cuadrado con una longitud de lado cualquiera. Al principio, un contador se inicializa en cero, se pasa, a continuación, cuatro veces por el mismo bucle. Cada vez se incrementa el contador en 1.

El comando REPITE (REPEAT en inglés) permite evitar estas repeticiones fastidiosas.

Escribiremos sencillamente:

```
PARA CUADRADO :LADO
REPITE 4 [AV :LADO DR 9Ø]
FIN
```

2. LA VISUALIZACION

Ya hemos hablado de los comandos RESPUESTA (OUTPUT en inglés), ESCRIBE (PRINT en inglés) y MUESTRA (TYPE en inglés).

Este último tiene la sintaxis siguiente:

MUESTRA objeto1 objeto2... objeton, contrariamente a ESCRIBE, permite visualizar objeto1... objeton, los unos al lado de los otros, en una misma línea.

Ejemplo:

```
?MUESTRA "ESTO MUESTRA [ES UNA PRUEBA]
ESTOES UNA PRUEBA?
?
?ESCRIBE "ESTO ESCRIBE [ES UNA PRUEBA]
ESTO
ES UNA PRUEBA
```

aquí objeto1 es la palabra "ESTO y objeto2 es la lista [ES UNA PRUEBA].

3. LOS FICHEROS

3.1. Generalidades

Es posible utilizar un disco como medio de almacenamiento de programas LOGO. No vamos a describir aquí en detalle el DOS (sistema de explotación de discos) para cada máquina en que LOGO esté implantado. Para ello diríjanse Uds. al manual de referencia de su micro-ordenador.

Describiremos solamente algunos puntos importantes que lo diferencian esencialmente de un lenguaje como BASIC.

Con Basic, en efecto, sólo es posible salvaguardar en disco o programas o datos previamente ordenados, en ficheros de tipo secuencial, directo o texto.

Cuando trabaja Ud. con LOGO, éste pone a su disposición un espacio de trabajo. Todo lo que va Ud. a definir: variables, procedimientos, etc., está almacenado en ese espacio. Si efectúa Ud. una salvaguardia en disco se salvaguarda el espacio entero, es decir, los procedimientos, las variables y sus valores.

Notemos, igualmente, que en ciertas máquinas (Apple II, IIe entre otras) es posible salvaguardar imágenes binarias en disco.

3.2. La sintaxis

Haremos una breve descripción de la sintaxis que comportan todos los comandos clásicos que ya conocen.

GUARDA:

GUARDA "nombre de fichero (en inglés SAVE) permite salvaguardar todo el espacio de trabajo.

RECUERDA:

RECUERDA "nombre de fichero (en inglés LOAD o READ) es el comando inverso de GUARDA: carga en memoria el espacio de trabajo.

CATALOGA:

(CATALOG en inglés) permite listar la biblioteca de los ficheros contenidos en el disco.

DESTRUYE:

(ERASE en inglés) permite borrar un fichero no deseado del disco.

Hoy día existe también la posibilidad de almacenar y leer imágenes binarias en disco. La sintaxis es la siguiente:

READPICT "nombre de fichero permite ir a leer una imagen en disco y, al mismo tiempo, cargar el espacio de trabajo. (Ve Ud. formarse la imagen y aparecer en pantalla).

SAVEPICT "nombre de fichero, permite, por el contrario, efectuar más o menos un "hardcopy" de pantalla y almacenar ésta última en disco.

ERASEPICT "nombre de fichero permite, finalmente, borrar una imagen binaria almacenada en disco.

Ya hemos terminado con este capítulo.

Esperamos que estas nociones, más detalladas ahora, les ayudarán a abordar más fácilmente el próximo capítulo, en el que hemos desarrollado un cierto número de programas.

7

Programar en LOGO

1. INTRODUCCION

En este capítulo vamos a desarrollar un cierto número de programas, que son, más o menos, complemento de lo que ya hemos visto hasta aquí. Así, ciertos procedimientos matemáticos completan los primitivos LOGO que tratan los números de los que ya hablamos en el Capítulo IV.

Algunos programas gráficos, basados esencialmente en nociones de geometría elemental, y algunos programas interactivos un poco más consecuentes sobre listas y cadenas de caracteres, acaban este capítulo.

2. ALGUNOS PROCEDIMIENTOS ARITMETICOS

En el capítulo IV describimos un cierto número de procedimientos muy simples, corrientemente utilizados, entre los que se pueden citar PRODUCTO, RESTO, COCIENTE, ENT (parte entera)... que permiten hacer un mínimo de cálculos en trigonometría. Añadiendo tg X, arcsen X, arccos X a las funciones ya residentes, se acrecienta sensiblemente la biblioteca de funciones trigonométricas.

Para hacer otros programas un poco más consecuentes, puede ser interesante tener a su disposición un repertorio de procedimientos más

amplio. Entre los más útiles figuran especialmente el cálculo del cuadrado de un número, su cubo, etc...

El procedimiento CUADRADO siguiente permite calcular el cuadrado de un número real X cualquiera.

```
PARA CUADRADO :X
RESPUESTA :X * :X
FIN

?ESCRIBE CUADRADO 7
49
?
?
?ESCRIBE CUADRADO 56
3136
?
?
?ESCRIBE CUADRADO 3.897
15.1866
```

Del mismo modo, el procedimiento CUBO calcula a partir de CUADRADO el cubo de un número cualquiera.

En efecto: $X^3 = X^2 * X$ y efectuamos X^2 introduciendo en el teclado CUADRADO X.

```
PARA CUBO :X
RESPUESTA :X * CUADRADO :X
FIN

?ESCRIBE CUBO 6.54
279.726
?ESCRIBE CUBO 3.Ø
27.

?ESCRIBE CUBO 4
64
?ESCRIBE CUBO –3.674
–49.5927
```

Por recurrencia, $X^n = X^{n-1} * X$

$X^{n-1} = X^{n-2} * X$
etc...

hasta:

$X^2 = X * X$
$X = X^1 * X^0 = X * 1 = X$

El procedimiento POTENCIA, que permite calcular las potencias enteras positivas de un número real, se define así:

```
PARA POTENCIA :X :N
SI :N = 1 [RESPUESTA :X]
DA "N :N – 1
RESPUESTA :X * POTENCIA :X :N
FIN

?ESCRIBE POTENCIA 4 3
64
?ESCRIBE POTENCIA 3 3
27
```

Vemos que, naturalmente, POTENCIA 2:X es equivalente a CUADRADO :X y que POTENCIA 3:X efectúa la misma operación que CUBO :X.

Otro procedimiento muy útil es el procedimiento INV, que calcula el inverso de un número real no nulo.

```
PARA INV :X
RESPUESTA / 1 :X
FIN

?ESCRIBE INV 2
.5
?ESCRIBE INV 5
.2
```

A partir de ello, se puede, igualmente, extender el cálculo del procedimiento POTENCIA *a* los exponentes negativos.

Así, para calcular X^{-2}, es suficiente con calcular X^2 y luego tomar su inversa (pues X^{-1} es también igual a $1/X$).

NPOTENCIA, que permite calcular las potencias negativas de X, se define así:

```
PARA NPOT :X :N
INV POTENCIA :X – :N
FIN

?ESCRIBE NPOT 2 – 3
.125
?ESCRIBE NPOT 7 – 9
2.478Ø9N8
```

Finalmente, para calcular las potencias tanto positivas como negativas, es suficiente con definir el procedimiento TPOTENCIA.

```
PARA TPOT :X :N
SI :N < Ø [NPOT :X :N] [POTENCIA :X :N]
FIN

?ESCRIBE TPOT 4 5
1Ø24

?ESCRIBE TPOT 4 – 8
1.52588N5
?ESCRIBE TPOT 3 –3
3.7Ø37N2
```

3. UN PROGRAMA DE CAMBIO DE BASE

La noción de base les es probablemente familiar: la base más corrientemente utilizada es, desde luego, la base 1Ø, pero aquellos de Uds. que estén familiarizados con el lenguaje ensamblador han oído, ciertamente, hablar de la base 2, de la base 8 y de la base 16.

En base 2, las únicas cifras autorizadas son el Ø y el 1. Para convertir 11 en base 2, se efectúan tantas divisiones entre 2 como sea posible. Así:

$$11 = 2 * 5 + 1$$
$$5 = 2 * 2 + 1$$
$$2 = 2 * 1 + Ø$$

o sea, $11 = 1 * 2^3 + Ø * 2^2 + 1 * 2^1 + 1 * 2^0$

11 se escribe 1Ø11 en base 2.

Del mismo modo para la base 8: expresemos 1ØØ en base 8:

$$1ØØ = 8 * 12 + 4$$
$$12 = 8 * 1 + 4$$
$$1ØØ = 1 * 8^2 + 4 * 8 + 4^1 * 8^0$$
$$= 144 \text{ en base 8.}$$

La cifra de la derecha es el resto de la división de 1ØØ por 8. El penúltimo es el resto de la división de 12 por 8, siendo 12 el cociente de 1ØØ por 8 y 4 el resto de la división de 1ØØ por 8.

Escribamos un procedimiento que permita hacer la conversión a base 5 de un número escrito en base 1Ø:

```
PARA BAS5 :N
SI :N = Ø [RESPUESTA Ø]
RESPUESTA PALABRA BAS5 COCIENTE :N 5 RESTO :N 5
FIN

?ESCRIBE BAS5 5
Ø1Ø
?ESCRIBE BAS5 8
Ø13
?ESCRIBE BAS5 78
Ø3Ø3
```

Un procedimiento más general permite convertir un número N en una base cualquiera M.

```
PARA BAS :M :N
SI :N = Ø [RESPUESTA Ø]
RESPUESTA PALABRA BAS :M COCIENTE :N :M RESTO :N :M
FIN
```

```
?ESCRIBE BAS 8 1Ø
Ø12
?ESCRIBE BAS 8 16
Ø2Ø
?ESCRIBE BAS 3 4
Ø11
?ESCRIBE BAS 2 1ØØ
Ø1ØØ1ØØ
?ESCRIBE BAS 8 1ØØ
Ø144
```

Habrán notado, sin duda, que el procedimiento precedente tiene el inconveniente de añadir sistemáticamente un cero al comienzo del resultado.

Podemos evitar esto modificando ligeramente el procedimiento BASE.

```
PARA BASE :M :N
SI :N = Ø [RESPUESTA Ø]
RESPUESTA SP BAS :M :N
FIN

?ESCRIBE BASE 2 3
11
?ESCRIBE BASE 8 16
2Ø
```

Se constata que el cero en cuestión ha sido suprimido.

Queda por presentar una última mejora: han debido Uds. observar que el procedimiento BASE sólo es utilizable para bases menores de 1Ø. Para las otras bases es preciso utilizar nuevos símbolos para representar los números 11, 12, 13, etc...

En base 16, por ejemplo, se representa 1Ø por A, 11 por B, 12 por C, 13 por D, 14 por E, 15 por F.

La modificación a efectuar es verdaderamente mínima, pues es suficiente, en efecto, con añadir algunos tests en BASE para dar A ó B, ó C en lugar respectivamente de 1Ø, 11, 12 cuando ello es necesario.

Cuando, por ejemplo, nos encontremos un resto igual a 13, el resultado será D.

Dejaremos que hagan Uds. solos esta modificación mínima para pasar a la descripción del programa siguiente.

4. ALGUNOS DIBUJOS

En el capítulo 5, reservado a la descripción de la tortuga gráfica, hemos desarrollado un cierto número de programas bastante simples, basados, principalmente, en trazado de círculos, triángulos y poliedros.

Aquí nos proponemos realizar algunos dibujos muy simples en los que los conocimientos matemáticos exigidos son mínimos.

Los dos dibujos realizados aquí están basados en el trazado de un arco de círculo.

En el capítulo 5 vimos cómo trazar un círculo con el procedimiento CIRCULO1.

```
PARA CIRCULO1 : RA1
REPITE 360 [AV : RA1 DR 1]
FIN
```

De la misma forma, podemos trazar un arco de círculo cuyo ángulo al centro y radio podemos elegir.

Para ello, definimos un procedimiento calculado de CIRCULO1:

```
PARA AR :MEDIDA :ANGULO
REPITE ANGULO [AV :MEDIDA DR 1]
FIN
```

Luego, del mismo modo que hemos definido CIRCULO a partir de CIRCULO 1, escribiremos el procedimiento ARCO utilizando el procedimiento AR.

```
PARA ARCO :RADIO :ANGULO
AR :RADIO * 1.74N2 :ANGULO
FIN
```

Así, para trazar un arco de círculo con un ángulo en el centro igual a 90 grados y de radio igual a 10, introduciremos en el teclado:

```
ARCO 10 90
```

Dibujemos ahora una silueta de pájaro utilizando arcos y círculos:

```
PARA PAJARO :LADO
ARCO :LADO 70
FRUMBO 20
ARCO :LADO 70
FRUMBO 0
FIN
```

Para dibujar varios, escribiremos otro procedimiento basado en los procedimientos precedentes AR, ARCO, PAJARO:

```
PARA CAMPO
LL FPOS [-110 50] IZ 10
REPITE 4 [BL PAJARO 40 LL DR 140 AV 25]
FIN
```

Dibujemos ahora tres bonitas flores.

Es preciso, en primer lugar, dibujar un pétalo, luego una flor completa. Para ello hemos definido el procedimiento PETALO siguiente:

```
PARA PETALO :LADO
ARCO :LADO 60
DR 120
ARCO :LADO 60
DR 120
FIN
```

Luego el procedimiento FLOR:

```
PARA FLOR :MEDIDA :ANGULO
REPITE COCIENTE 360 :ANGULO [PETALO :MEDIDA DR :ANGULO] FRUMBO
180
AV 40 LL RE 40
FIN
```

el número de pétalos es variable, en consecuencia, debe ser especificado por el usuario. El trazado de los diversos pétalos corresponde a las líneas de comienzo del procedimiento, correspondiendo el dibujo del tallo a las instrucciones:

```
FRUMBO 180 AV 40 LL RE 40
```

Para dibujar tres flores, separadas unas de otras por un espacio correcto, utilizaremos tres veces consecutivas el procedimiento FLOR escribiendo un nuevo procedimiento:

```
PARA PFLOR
LL FPOS [-70 0] BL
REPITE 3 [FLOR 30 30 LL FRUMBO 90 AV 60 FRUMBO 0 BL]
FIN
```

5. ALGUNOS EJEMPLOS DE PROGRAMAS INTERACTIVOS

En primer lugar, ¿está Ud. fuerte en cálculo mental? He aquí un programa que le permitirá poner a prueba sus capacidades.

Introduzca en el teclado la respuesta a la pregunta que se le plantea. Si no introduce Ud. un número, el programa lo detecta y se para. Finalmente, indica si su respuesta es correcta o no.

Introduzcan por el teclado el procedimiento siguiente:

```
PARA CAL
NOMBRA ALEATORIO 100 "N1
NOMBRA ALEATORIO 100 "N2
NOMBRA :N1 + :N2 "N0
ESCRIBE F :N1 F [ + ] :N2
DA "N LL
TEST NUMEROP PRIMERO :N
SIF [RESPUESTA [INTRODUZCA UN NUMERO!] ]
SIC [TEST :N0 = PRIMERO :N]
SIF [RESPUESTA "FALSO]
SIC [RESPUESTA "CIERTO]
FIN
```

y hagan Uds. ejercicios...

```
?ESCRIBE CAL
72 + 97
179
FALSO

?ESCRIBE CAL
55 + 69
TY
INTRODUZCA UN NUMERO!

?ESCRIBE CAL
4 + 8
12
CIERTO
```

Realicemos ahora algunos programas interactivos que manipulen palabras y listas.

En primer lugar, puede ser útil disponer de un procedimiento que permita contar el número de elementos de una lista. Esta operación puede definirse recursivamente de una forma muy sencilla:

– Si la lista es igual a una lista vacía, la longitud de la lista es convencionalmente igual a Ø.

– En caso contrario, es igual a 1 añadido a la longitud de la lista privada de su primer elemento.

Escribiremos pues:

```
PARA LONG :X
SI :X = [ ]  [RESPUESTA Ø]
RESPUESTA 1 + LONG SP :X
FIN
```

Ejemplo:

```
?LONG [I OUT J]
NO SE QUE HACER CON 3
?
?
?ESCRIBE LONG [R T ER J]
4
?ESCRIBE LONG [ ]
Ø
```

Del mismo modo que el procedimiento ALEATORIO permite generar aleatoriamente un número, podemos, igualmente, crear un generador de palabras aleatorias.

Para ello, tenemos necesidad de definir previamente otro procedimiento. Tengamos una lista X y un número entero N dados, el procedimiento PIC tiene el papel de dar el eNésimo elemento de la lista. Este procedimiento recursivo opera de la forma siguiente:

```
PARA PIC :N :X
SI :N = 1 [RESPUESTA PRIMERO :X]
RESPUESTA PIC – :N 1 SP :X
FIN
```

Ejemplo:

```
?ESCRIBE PIC 1 [HJ KL IOY]
HJ
?ESCRIBE PIC 2 [MA DF ER]
DF
```

Ahora ya podemos, fácilmente, generar aleatoriamente un elemento de una lista cualquiera X utilizando conjuntamente los procedimientos PIC y LONG.

LONG :X da el número de elementos N de la lista :X.

ALEAT LONG :X proporcionará un número comprendido entre Ø y N - 1.

Como debemos aplicar PIC a un número no nulo, definiremos PICALEAT de la forma que sigue:

```
PARA PICALEAT :X
RESPUESTA PIC + 1 ALEAT LONG :X :X
FIN
```

Ejemplo:

```
?ESCRIBE PICALEAT [MA DF ER T]
DF

?ESCRIBE PICALEAT [KLEBER MOZART DESCARTES ALVAREZ]
ALVAREZ
?ESCRIBE PICALEAT [DF MEJKL]
MEJKL
```

Gracias a estos procedimientos generales podemos programar un juego que quizá Uds. conozcan bajo el nombre de "DE LA HABANA VENIA UN BARCO CARGADO DE ... " Se da la letra del alfabeto y hay que encontrar un nombre, una ciudad, un vegetal, un animal, etc., que empiecen por esa letra.

Hemos adoptado cuatro categorías de objetos a encontrar: nombres, ciudades, vegetales y apellidos.

El procedimiento APRENDE siguiente tiene como objeto dar a la máquina ciertos conocimientos previos.

```
PARA APRENDE
DA "APELLIDOS [KLEBER MOZART DESCARTES ALVAREZ]
DA "NOMBRES [ALICIA BERNARDO CLAUDIO DOMINICA
FRANCISCO MARIA]
DA "CIUDADES [PARIS BONN MADRID LONDRES BANGKOK]
DA "VEGETALES [PIÑA MANZANA COL LECHUGA]
FIN
```

Ejemplo:

```
VEGETALES ES [PIÑA MANZANA COL LECHUGA]
CIUDADES ES [PARIS BONN MADRID LONDRES BANGKOK]
NOMBRES ES [ALICIA BERNARDO CLAUDIO DOMINICA
FRANCISCO MARIA]
APELLIDOS ES [KLEBER MOZART DESCARTES ALVAREZ]
```

En esta etapa el microordenador ya conoce un cierto número de ciudades, nombres, vegetales y apellidos.

Para enseñarle más, podremos definir un procedimiento de aprendizaje para cada tipo de objeto. Por ejemplo, para enseñarle un nuevo nombre de vegetal se definirá el procedimiento APVE:

```
PARA APVE :X
DA "VEGETALES F :VEGETALES :X
FIN
```

Ejemplo:

```
?APVE "ZANAHORIAS
?:VEGETALES
NO SE QUE HACER CON [PIÑA MANZANA COL LECHUGA
ZANAHORIAS]
```

Del mismo modo para las ciudades:

```
PARA APCI :X
DA "CIUDADES F :CIUDADES :X
FIN

?APCI "FRANCFORT
?.CIUDADES
NO SE QUE HACER CON [PARIS BONN MADRID LONDRES
BANGKOK FRANCFORT]
```

Los nombres:

```
PARA APNO :X
DA "NOMBRES F :NOMBRES :X
FIN
```

Ejemplo:

```
?APNO "IVAN
?ESCRIBE :NOMBRES
ALICIA BERNARDO CLAUDIO DOMINICA FRANCISCO
MARIA IVAN
```

Y los apellidos:

```
PARA APAP :X
DA "APELLIDOS F :APELLIDOS :X
FIN
```

Ejemplo:

```
?APAP "GUERRA
?:APELLIDOS
NO SE QUE HACER CON [KLEBER MOZART DESCARTES
ALVAREZ GUERRA]
```

Pasemos al juego propiamente dicho. Hemos escrito el procedimiento BARCO siguiente:

```
PARA BARCO
ESCRIBE [INTRODUZCA LA LETRA DE COMIENZO]
DA "L LEECAR
ESCRIBE [NOMBRES APELLIDOS CIUDADES O VEGETALES?]
DA "LI PRIMERO LEELISTA
DA "X COSA :LI
LABEL "BUCLE
DA "M PICALEAT :X
SI IGUALP :L PRIMERO :M [RESPUESTA :M STOP]
DA "X SP :X
SI: X = [ ] [RESPUESTA [ESTOY PEZ] ]
VE "BUCLE
FIN
```

Veamos un poco cómo funciona este procedimiento: utiliza los primitivos LEECAR, LEELISTA, COSA Y DA que ya vimos en capítulos precedentes, así como el procedimiento PICALEAT que acabamos de ver.

Recordemos que LEECAR permite introducir un caracter en el teclado sin que aparezca en pantalla. LEELISTA es el procedimiento idéntico para las listas.

COSA Y DA se utilizan para nombrar los datos.

Para jugar con la máquina, escoja en primer lugar una letra para empezar; introdúzcala por el teclado cuando vea el mensaje:

"INTRODUZCA LA LETRA DE COMIENZO"

el programa la almacena bajo el nombre "L".

Escoja ahora la categoría de objetos a encontrar: introduzca por teclado una de las listas siguientes: "APELLIDOS", "NOMBRES", "CIUDADES", "VEGETALES"...

Esto se almacena en "LI".

En "X", almacenamos a continuación el contenido de la variable "LI". Al comienzo del programa "X" es, pues, igual a una de las listas siguientes:

KLEBER MOZART DESCARTES ALVAREZ
ALICIA BERNARDO CLAUDIO DOMINICA FRANCISCO MARIA
PARIS BONN MADRID LONDRES BANGKOK
PIÑA MANZANA COL LECHUGA

A continuación, uno de los elementos es tomado al azar en esta lista "X" gracias al procedimiento PICALEAT; luego es salvaguardado en "M".

Si comienza por la letra "L", el programa muestra la palabra en cuestión en pantalla.

En el caso contrario, se vuelve a aplicar PICALEAT a la lista "X" privada de su primer elemento y así consecutivamente hasta que la lista "X" esté vacía... y hasta que su máquina diga:

"ESTOY PEZ"

Ejemplo:

?ESCRIBE BARCO
INTRODUZCA LA LETRA DE COMIENZO
APELLIDOS NOMBRES CIUDADES O VEGETALES?

```
CIUDADES
ESTOY PEZ
?ESCRIBE BARCO
INTRODUZCA LA LETRA DE COMIENZO
APELLIDOS NOMBRES CIUDADES O VEGETALES?
APELLIDOS
MOZART
```

Desde luego, sería posible aportar diversas mejoras a este programa para efectuar una partida de juego completa, en el curso de la cual el usuario (es decir, Ud. mismo) y su máquina se enfrentarían.

Cada uno debería dar obligatoriamente un nombre de cada categoría: apellidos, nombres, ciudades, vegetales.

Sería preciso también preveer cómo efectuar la cuenta de los puntos marcados por los dos jugadores.

Podremos adoptar la norma convencional siguiente:

Cada jugador puntúa:

- Dos puntos si los dos jugadores encuentran palabras diferentes.
- Un solo punto si el otro jugador ha encontrado una palabra idéntica.

No se puntúa en ningún otro caso.

Les dejamos el esfuerzo de programar este procedimiento y de imaginar otras posibles mejoras...

8

Glosario de los primitivos LOGO

Este capítulo va a examinar brevemente los primitivos de los que ya hemos hablado en este libro, pero, esta vez, considerando su implementación sobre tal o cual máquina. Los primitivos están descritos siguiendo el orden de aparición en los capítulos precedentes. Daremos aquí el equivalente para cada máquina de la denominación española que hemos adoptado.

A. LAS DIVERSAS IMPLEMENTACIONES DE LOGO

Pasemos revista, en primer lugar, a las máquinas en que LOGO está implementado.

Hoy día, sólo están comercializadas versiones de LOGO en APPLE y TI 99. LOGO está anunciado en el microordenador francés TO 7, el ZX Spectrum, el Commodore 64 así como en el IBM PC.

LOGO en APPLE II y APPLE IIe ha sido escrito por LOGO Computer Systems. Es distribuido por Apple Computer Company Inc. bajo el nombre de APPLE LOGO. Necesita un APPLE II 64 K (una tarjeta de lenguaje) o un APPLE II E, así como un drive, pues el lenguaje está contenido en disquete.

Ese LOGO es muy completo: dispone, desde luego, de la tortuga gráfica, de procedimientos potentes de manipulación de palabras y listas, de aritmética de coma flotante y de un sistema de explotación

de ficheros. El acceso al lenguaje máquina y a rutinas ensamblador es igualmente posible.

La misma sociedad, LOGO Computer Systems, anuncia una versión de LOGO para el IBM PC, basada en los mismos principios y las mismas posibilidades.

El lenguaje está también disponible en disquete y necesita un IBM PC con 128 K de memoria.

LOGO ha sido implementado en el TI 99 de Texas Instruments.

Como en el Apple II, existe una tortuga gráfica, posibilidades casi idénticas de manipulación de palabras y de listas, así como un sistema de explotación de ficheros.

Sin embargo, el TI 99, por su configuración material, posee animacion dinámica real: se pueden utilizar fácilmente "sprites" (en español "duendes"), cuya forma, color y velocidad se pueden programar... y que son muy conocidos gracias a los videojuegos.

Desgraciadamente, el TI 99 en LOGO sólo trabaja con enteros, lo que reduce mucho las posibilidades matemáticas del lenguaje.

En el apartado siguiente, volveremos sobre cada capítulo desarrollado en esta obra y comentaremos poco a poco los procedimientos que hemos introducido.

No obstante, no tenemos intención de comentar todos los primitivos propios de cada máquina, ni de describir con detalle su editor de texto o su sistema de explotación de ficheros. Para ello, diríjanse Uds. al manual de referencia de su microordenador.

B. LA SINTAXIS LOGO

En este apartado sólo abordaremos los primitivos que varían cuando se pasa de una máquina a otra.

1. LOS PRIMITIVOS GRAFICOS

En el Apple (Apple II o Apple IIe), para pasar al modo gráfico a partir del modo texto, es suficiente con introducir por teclado un co-

mando gráfico cualquiera; entonces aparece la tortuga gráfica en pantalla.

En el TI 99, por el contrario, se debe utilizar el comando TELL TURTLE.

Las otras diferencias de sintaxis se detallan en el cuadro siguiente.

Hacemos figurar sobre una misma línea un primitivo propio de una máquina con su equivalente en la otra.

PROCEDIMIENTO	APPLE	TI 99
BORRA	CLEAN	
ORIGINA	HOME	CLEARSCREEN
BORRAPANTALLA	CLEARSCREEN	
LL (LEVANTALAPIZ)	PU (PENUP)	PENUP
BL (BAJALAPIZ)	PD (PENDOWN)	PENDOWN
LB (LAPIZBORRA)	PE (PENERASE)	PE
LI (LAPIZINVERSO)	PX (PENEVERSE)	PRV
FIJAX	SETX	SX
FIJAY	SETY	SY
FPOS	SETPOS	SXY
FIJALAPIZ	SETPC número	SETCOLOR:nombre color SETCOLOR Número
FCF	SETBG número	COLORBACKGROUND: nombre de color
HACIA	TOWARDS	
POS	POS	
RUMBO	HEADING	WHERE

En estas dos versiones, los comandos RIGHT, LEFT, BACK y FORWARD han sido implementados... así como otros primitivos que ya hemos mencionado y de los que no hablaremos aquí.

2. LOS NUMEROS

Tanto en el Apple como en el TI 99 existen primitivos de cálculo tales como +, -, *, /.

La diferencia esencial entre la versión del Apple y la del TI 99 es que la primera trabaja en coma flotante (con notaciones entera, decimal y exponencial) mientras que la segunda sólo trabaja con enteros.

El TI 99 trabaja generalmente en notación infijada, sin embargo, les será muy fácil escribir sus propios procedimientos prefijados.

Además de la limitación que acabamos de indicar, muchos primitivos presentes en la versión Apple tales como ROUND (que hemos introducido bajo el nombre REDONDEA), REMAINDER (RESTO), COS, SIN INTEGER, SQRT no están implantados en el TI 99.

El primitivo RANDOM (que hemos llamado ALEATORIO) existe en las dos máquinas:

En el Apple la sintaxis es la siguiente:

RANDOM numero

y el resultado es un número aleatorio comprendido entre Ø y "número"-1.

En el TI 99, RANDOM no tiene entrada y genera sistemáticamente un número entero comprendido entre 0 y 9.

En el Apple, el primitivo RANDOMIZE permite cambiar la secuencia de números que crea RANDOM.

En el TI 99 no existe equivalente de RANDOMIZE.

3. LAS PALABRAS Y LAS LISTAS

Hay pocas diferencias en cuanto a la sintáxis de las palabras y las listas: encontramos en los dos casos los primitivos LIST, SENTENCE, FIRST, BUTFIRST, LAST, BUTLAST...

La mayor diferencia entre el Apple y el TI 99 es que los números no son considerados como palabras en el TI 99.

Así, todos los primitivos de manipulación de palabras tales como PALABRA, PRIMERO, SALVOPRIMERO, ULTIMO, SALVOULTIMO (que existen en estas dos máquinas bajo los nombres respectivos de WORD, FIRST, BUTFIRST, LAST, BUTLAST) en el TI 99 no se aplican a los números.

Para manipular un número es preciso hacer que vaya precidido de comillas.

Así: FIRST 25 es incorrecto

mientras que: FIRST "25 tiene como resultado 2

Además no hay palabra vacía en el TI LOGO.

Si Ud. introduce en el teclado:

```
PRINT"
```

verá como aparece en pantalla un error de sintaxis.

El primitivo LEELISTA se denomina READLIST en el Apple y READLINE en el TI 99.

El primitivo LEECAR se denomina READCAR en las dos máquinas.

Por otro lado, en el TI 99, la mayoría de los primitivos no pueden tener más de dos entradas.

Debemos, por ejemplo, escribir:

```
PRINT WORD "A WORD "B "C
```

para obtener la palabra "ABC".

Entre los primitivos que acabamos de ver, sólo SENTENCE puede tener varias entradas.

4. PROCEDIMIENTOS Y VARIABLES

En las dos máquinas, un procedimiento se define con la ayuda de TO y de END.

En el interior de un procedimiento, es posible realizar bifurcaciones gracias a una expresión del tipo:

SI predicado <expresión1> <expresión2>

o

SI predicado <expresión>

Para el Apple, se adoptará la sintaxis siguiente:

IF predicado <exp1> <exp2>

o

IF predicado <exp>

y para el TI 99, tendremos solamente:

IF predicado THEN <expresión>.

Para nombrar las variables se utiliza el comando MAKE (en español DA o HAZ) en las dos máquinas —así como el comando NOMBRA (NAME en el Apple y CALL en el TI 99).

5. OTROS PROCEDIMIENTOS

Reagrupemos en un mismo cuadro las equivalencias de los restantes procedimientos:

PROCEDIMIENTO	APPLE	TI 99
Y	AND	BOTH
O	OR	EITHER
ASCII	ASCII	CHARNUM
CAR	CHAR	PRINTCHAR
CURSOR	CURSOR	
FIJACURSOR	SETCURSOR	PUTTITLE
MUESTRA	TYPE	TYPE
DA	MAKE	MAKE
NOMBRA	NAME	CALL

Indiquemos también que es posible salvaguardar en un fichero el espacio de trabajo, e igualmente, ir a leer un fichero y cargarlo en memoria.

Esto se efectúa gracias a comandos del tipo GUARDA (SAVE en las dos máquinas) y RECUERDA (LOAD en el Apple y RECALL en el TI 99).

Pero, para más detalles, les aconsejamos se dirijan al manual técnico de su microordenador.

Otros libros sobre lenguajes informáticos publicados por

PARANINFO S.A.

BASIC. Programación de microordenadores. *Checroun. 4ª edición. 112 páginas.*

Para quienes carecen de conocimientos sobre ordenadores. Elementos que constituyen el sistema de información en torno a un microordenador. Importancia del lenguaje de programación y sus niveles.

Programación con el lenguaje COBOL. *Galán Pascual. 2ª edición. 328 páginas.*

El COBOL como herramienta de trabajo. Imprescindible para quienes han de introducirse en la programación de ordenadores aplicados a la gestión empresarial.

BASIC. Introducción a la programación. *Larreche. 4ª edición. 132 páginas.*

Con numerosos ejercicios y ejemplos prácticos para aprender a escribir los propios programas y dar instrucciones al ordenador.

BASIC. Curso acelerado. *De Rossi. 3ª edición. 224 páginas.*

Para aprender con rapidez este lenguaje. Más de 350 ejemplos y ejercicios para practicar el BASIC en cualquier ordenador.

Programación con el lenguaje PASCAL. *Sanchis Llorca. 5ª edición. 368 páginas.*

Para que el lector pueda llegar a construir sus propios programas con calidad, buen estilo y fiabilidad. Numerosos ejercicios, ejemplos y programas completos.

BASIC para niños. Con notas didácticas para padres y educadores. *Watt y Mangada. 4ª edición. 128 páginas.*

Una obra eminentemente didáctica en la que los dibujos juegan un importante papel. Parte de conocimientos básicos sobre la utilización del ordenador y de conceptos elementales de programación BASIC. Un libro para los niños que deben conocer todos los padres y educadores.

De proxima aparición

BELLIDO.– Amaestra tu Dragón.
BELLIDO y SANCHEZ.– BASIC para maestros.
DELANNOY.– Ficheros en BASIC.
GARCIA MERAYO.– FORTRAN 77
HART.– Diccionario en BASIC.
MARSHALL.– Lenguajes de programación para micros: BASIC, PASCAL, LIPS, COBOL, FORTH, COMAL, FORTRAN, PILOT, PROLOG y LOGO.
MONTEIL.– Como programar su Commodore 64 (Basic - Gráficos - Sonido).
MONTEIL.– Como programar su Commodore 64 (Lenguajes máquina, entradas - salidas, periféricos).
PLOUIN.– IBM-PC (Características - Programación - Manejo).
WATT y MANGADA.– BASIC para niños con Dragón.
WATT y MANGADA.– BASIC avanzado para niños.

PRIMEROS PASOS EN LOGO

LOGO nació alrededor del año 1968 en los laboratorios de Inteligencia Artificial del MIT en los Estados Unidos. Posteriormente, ha sido utilizado con éxito en escuelas y universidades, y ahora está disponible en microordenadores tales como Apple II, Apple IIe, TI 99, Commodore 64 y el IBM-PC. Actualmente, anunciado en el ZX Spectrum, el TO 7 y otros, su difusión es mayor de día en día.

Por su concepción, es un lenguaje que resulta accesible para todos. Muy fácil, y rápidamente asimilable, abre las puertas de una programación cómoda, rica y muy completa.

LOGO posee, además, dos cualidades esenciales que hacen de él un excelente lenguaje de programación:

- permite la manipulación sin dificultad de cadenas de caracteres, texto y símbolos.
- posee un verdadero modo gráfico que permite la realización de dibujos muy sofisticados.

Esta obra se propone responder a las tres preguntas siguientes:

- ¿Qué es el LOGO?
- ¿Qué se puede hacer con LOGO?
- ¿Cómo programar en LOGO?

EDITORIAL PARANINFO SA
Magallanes, 25 - 28015 Madrid
ISBN: 84-283-1347-4